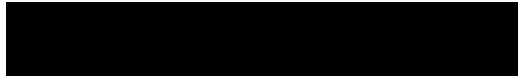


## **EXHIBIT 4**



**UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
MARSHALL DIVISION**

**TQ DELTA, LLC,**

*Plaintiff,*

v.

**COMMSCOPE HOLDING COMPANY, INC.,  
COMMSCOPE INC., ARRIS US HOLDINGS,  
INC., ARRIS SOLUTIONS, INC., ARRIS  
TECHNOLOGY, INC., and ARRIS  
ENTERPRISES, LLC**

*Defendants.*

CIV. A. NO. 2:21-CV-310-JRG  
(Lead Case)

---

**TQ DELTA, LLC,**

*Plaintiff,*

v.

**NOKIA CORP., NOKIA SOLUTIONS AND  
NETWORKS OY, and NOKIA OF AMERICA  
CORP.,**

*Defendants.*

CIV. A. NO. 2:21-CV-309-JRG  
(Member Case)

---

**OPENING EXPERT REPORT OF DR. RICHARD WESEL ON THE INVALIDITY  
OF THE ASSERTED CLAIMS OF THE FAMILY 3 PATENTS (U.S. PATENT NOS.  
7,844,882; 8,276,048; 8,495,473; 9,547,608)**

38. By at least October 12, 2004, I had qualifications that met or exceeded those of a POSA.

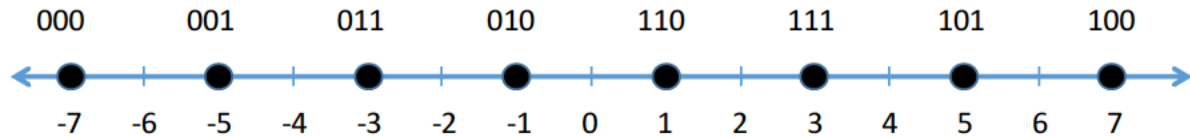
39. Counsel for Nokia has informed me that TQ Delta has proposed the following definition for a POSA: “A person of ordinary skill in the art at or before the time of the inventions of the Patents-in-Suit addressed in this declaration would have had a bachelor’s degree in electrical engineering with 2-3 years of experience in DSL communication systems. Alternatively, the person would have had a master’s degree in electrical engineering.” Cooklev Decl., dated March 14, 2022, ¶ 20. The opinions set forth in this report would not change under TQ Delta’s definition.

### **VIII. BACKGROUND OF THE TECHNOLOGY**

40. In this technical background, I describe the pertinent technology as it existed as of October 2004, the assumed priority date Family 3 Patents. As I will describe, the ideas of Reed-Solomon coding, interleaving, shared memory, and configuration messages that allow transceivers to exchange capabilities and agree on the parameters that define the communication link (including messages about maximum available interleaver memory) were all known to a POSA at the time of the alleged invention.

#### **A. Multicarrier Transmission**

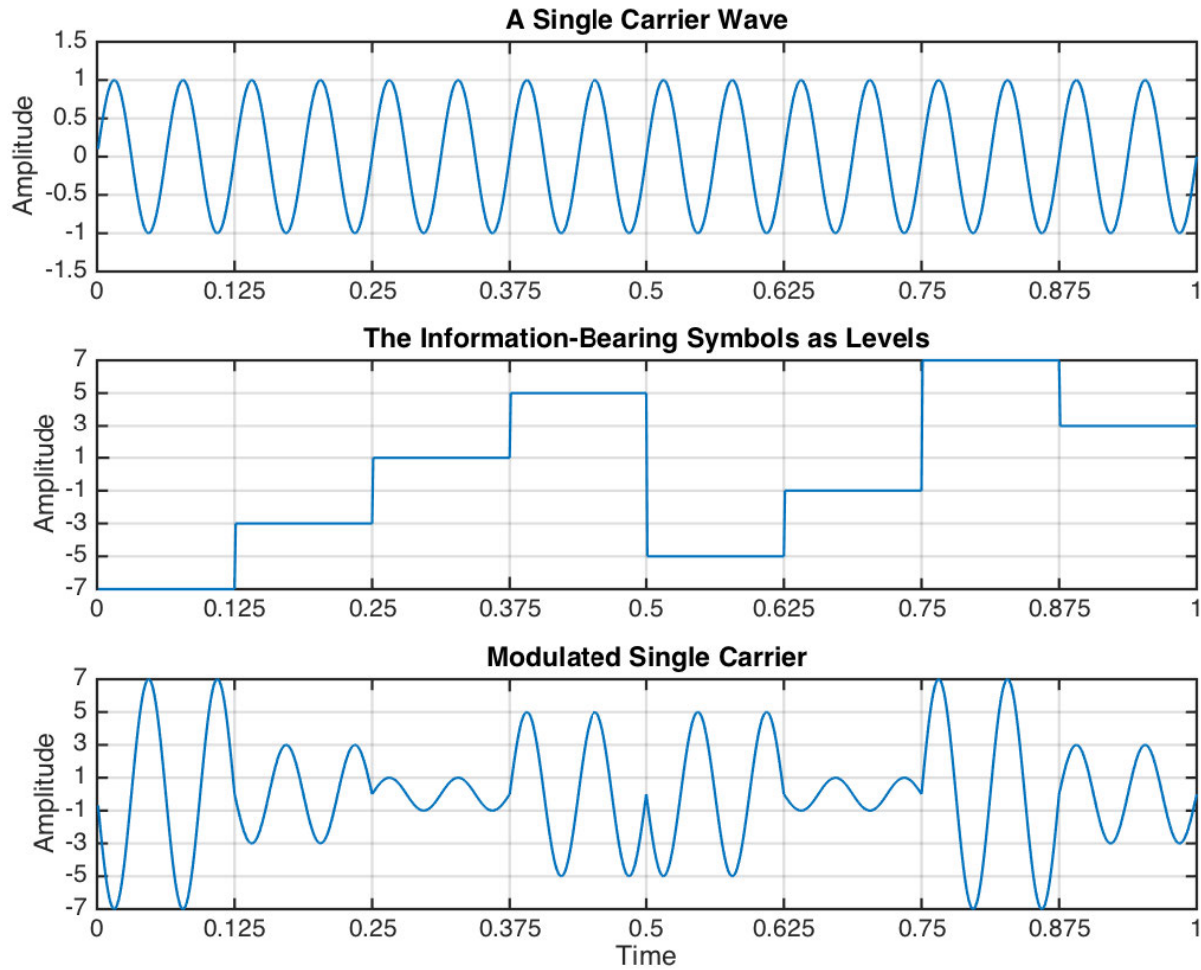
41. Information (bits) are often communicated using sinusoids, waveforms that look like the sine and cosine functions. Waveforms that carry information are called “carriers.” Through the process of modulation, a sine wave that is a carrier can carry information through its phase and amplitude. The discrete phases and amplitudes that communicate each possible bit pattern are represented by a constellation. For example, an 8 pulse amplitude modulation (8-PAM) constellation is shown below in Figure 1 with a binary reflected Gray code labeling.



**Figure 1: A constellation is symbols to transmit three bits per symbol using a binary reflected Gray labeling for 8 pulse amplitude modulation (8-PAM).**

42. Figure 2 below illustrates how a sinusoid can be used as the carrier wave to communicate information from the 8-PAM constellation of Figure 1. The top subfigure shows the carrier wave, which is a 16 Hz sine function. The middle subfigure shows the sequence of levels (-7, -3, 1, 5, -5, -1, 7, 3), which communicates the bits (000, 011, 110, 101, 001, 010, 100, 111) using the 8-PAM constellation with the labeling of Figure 1. The bottom subfigure shows the modulated single-carrier waveform produced by multiplying the waveforms in the top two subfigures.

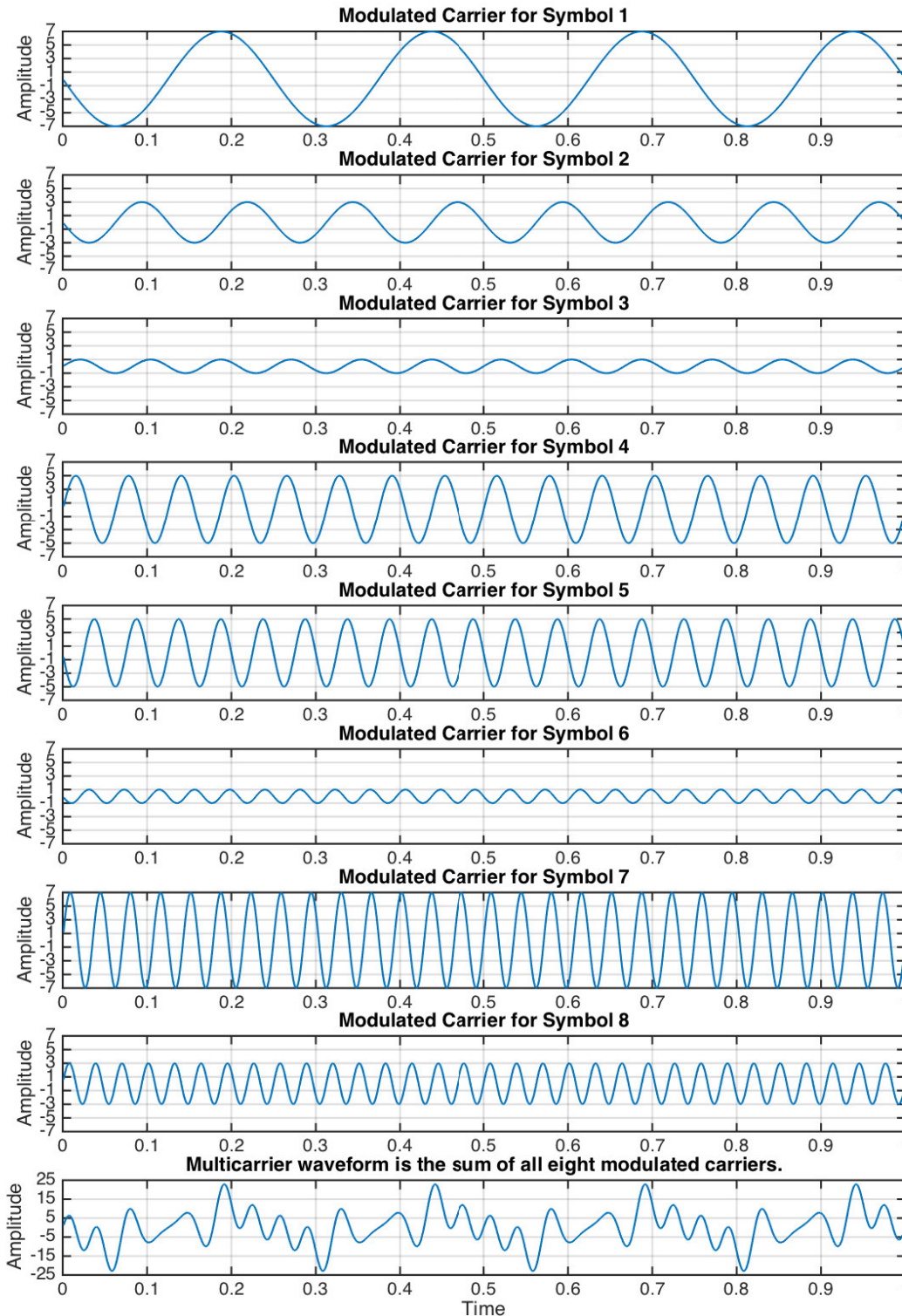




**Figure 2:** This figure illustrates single-carrier modulation of 8-PAM. The top subfigure shows the carrier wave, which is a 16 Hz sine function. The middle subfigure shows the sequence of levels (-7, -3, 1, 5, -5, -1, 7, 3), which communicates (000, 011, 110, 101, 001, 010, 100, 111) using the 8-PAM constellation with the labeling of 1. The bottom subfigure shows the modulated single-carrier waveform produced by multiplying the waveforms in the top two subfigures.

43. Notice that with single-carrier modulation each transmitted constellation point has its own symbol period where its associated level is the amplitude of the carrier wave. An alternative to single-carrier modulation is multicarrier modulation. Figure 3 below illustrates how multicarrier modulation can be used to communicate the same bit sequence as the example shown in Figure 2. The top eight subfigures each show a modulated sub-carrier sinusoid. The carriers

are modulated, respectively, by the levels (-7, -3, 1, 5, -5, -1, 7, 3), which communicate (000, 011, 110, 101, 001, 010, 100, 111) using the 8-PAM constellation with the labeling of Figure 1. These eight sub-carriers have frequencies of 4 Hz, 8 Hz, 12 Hz, 16 Hz, 20 Hz, 24 Hz, 28 Hz, and 32 Hz. The bottom subfigure shows the modulated multicarrier waveform produced by adding the waveforms in the top eight subfigures.



**Figure 3:** This figure illustrates multicarrier modulation. The top eight subfigures each show a modulated sub-carrier sinusoid, modulated, respectively, by the levels (-7, -3, 1, 5, -5, -1, 7, 3), which communicate (000, 011, 110, 101, 001, 010, 100, 111) respectively, using the 8-PAM constellation with the labeling of Error! Reference source not found.. These eight sub-c

**carriers have frequencies of 4 Hz, 8 Hz, 12 Hz, 16 Hz, 20 Hz, 24 Hz, 28 Hz, and 32 Hz. The bottom subfigure shows the modulated multicarrier waveform produced by adding the waveforms in the top eight subfigures.**

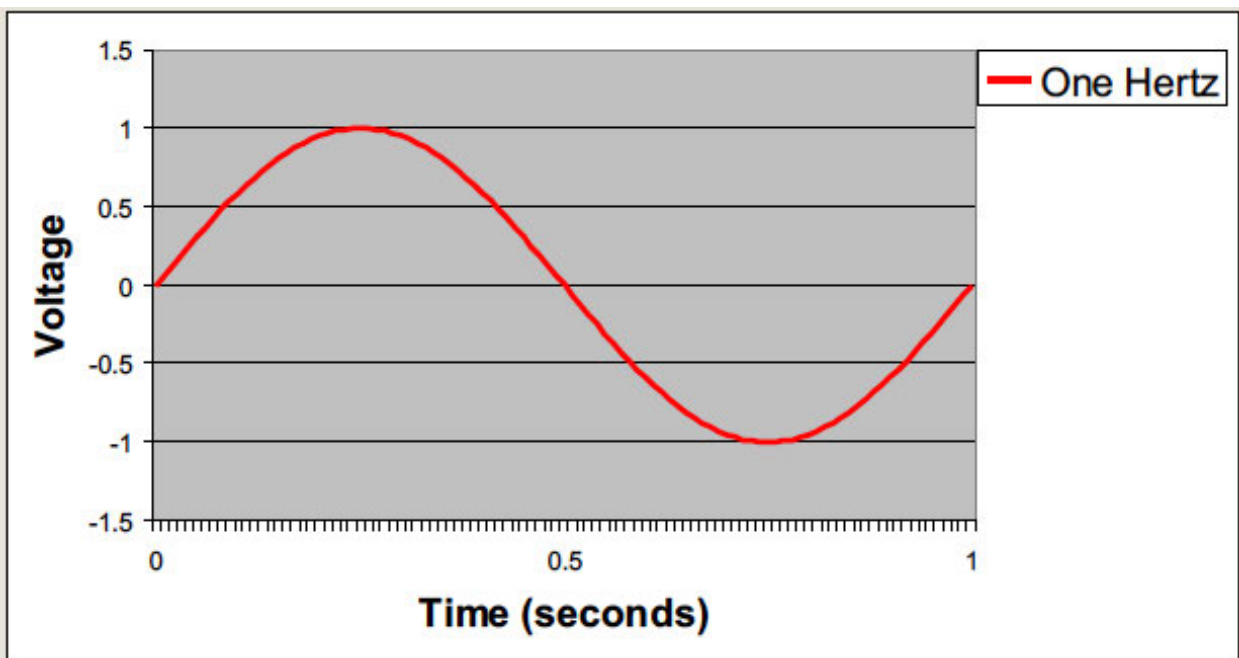
44. One way to think about the difference between multicarrier transmission and single-carrier transmission is that single-carrier transmission is analogous to a solo musician playing a melody on their instrument while multicarrier transmission would be analogous to an orchestra where the music from many individual instruments is transmitted simultaneously to the audience. During a symphony performance one might hear multiple melodies simultaneously and the human brain can focus on one of the melodies and tune out the others. With multicarrier transmission, there may be 64, 256, or 1024 “melodies” and the discrete Fourier transform is able to isolate the information from each “melody”, which is a sinusoidal carrier. Unlike a human ear and brain, a multicarrier transceiver can receive and distinguish the information from each of these “melodies” at the same time.

45. Another way to think about the difference between multicarrier transmission and single-carrier transmission is to think of single carrier transmission as a single-lane road in which one information-carrying vehicle at a time can move from the transmitter to the receiver. In contrast, multicarrier transmission is like a multi-lane highway where each sinusoidal carrier provides a separate “lane” on which information can travel.

46. One can think of the bandwidth of a communication channel as the overall width of roadway that can be used for either one very wide “lane” or multiple more narrow “lanes”. In this way one can understand that with a fixed amount of bandwidth, the total amount of information throughput with single-carrier and multicarrier is actually very similar. With single-carrier, the entire bandwidth is devoted to that one carrier. It is as if an extremely wide vehicle is traveling down the road with its “wide load” of information. In contrast, with multicarrier there can be many more “lanes” but since the lanes are more narrow, there is less information in each

lane. The overall information throughput is the same since both transportation systems are limited by the overall width of the roadway.

47. I now briefly describe how multicarrier modulation can be implemented using Fourier transforms. A sinusoid is sometimes categorized by its frequency expressed in units of Hertz (Hz), which indicates how many periods of the sinusoid are completed in one second. Figure 4 below shows a 1 Hz sine wave. Figure 5 below shows a 2 Hz sine wave, and Figure 6 shows a 10 Hz sine wave.



**Figure 4: A One Hertz sine wave.**

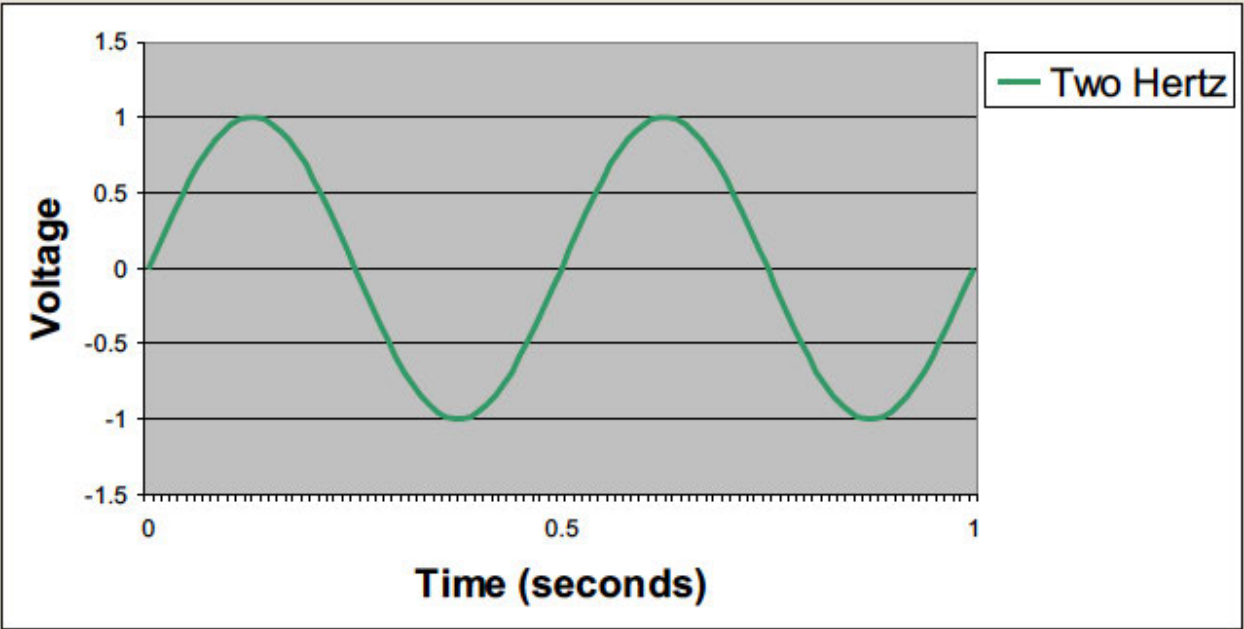


Figure 5: A two hertz sine wave

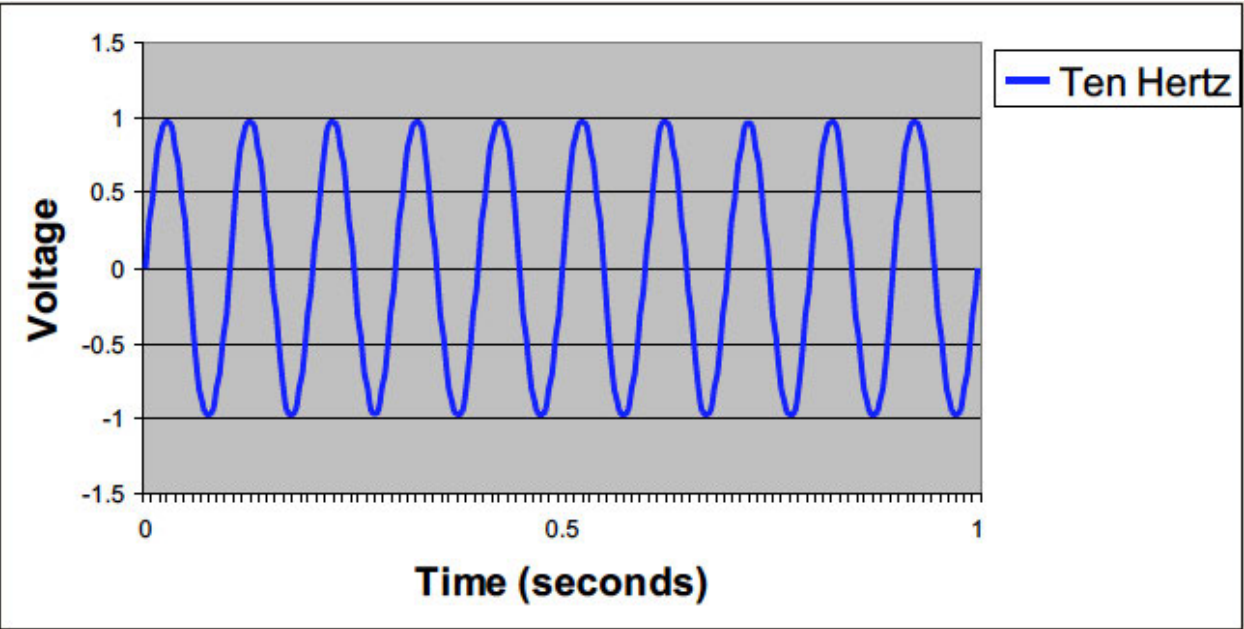
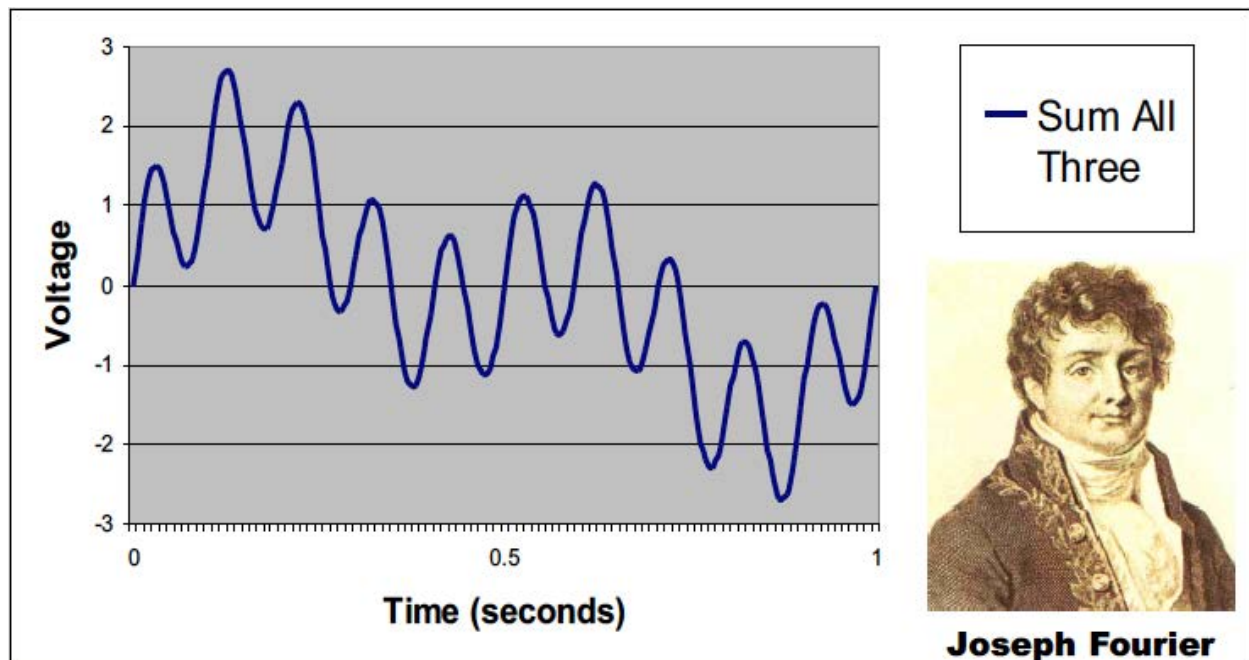


Figure 6: A ten Hertz sine wave.

48. Figure 7 shows one period of the periodic waveform produced by adding these three sinusoids. In 1807, Joseph Fourier proposed, and Dirichlet later proved rigorously, that an arbitrary periodic function can be represented by a sum of weighted sinusoids. See the discussion on page 371 of *Circuits, Signals, and Systems* by William Siebert and the English translation of *Théorie Analytique de la Chaleur* by Joseph Fourier.



**Figure 7: One period of the periodic waveform produced by adding 1 Hz, 2 Hz, and 10 Hz sine waves.**

49. There are a variety of Fourier relationships or “Fourier Transforms” that convert a time domain signal to its representation as a sum of sinusoids and back. Of particular interest in this case is the discrete Fourier transform (DFT) where a periodic sequence that repeats every  $N$  samples is represented as the sum of  $N$  complex sinusoids  $s_f(t) = \frac{1}{\sqrt{N}} e^{-\frac{j2\pi ft}{N}}$  where  $f, t \in$



$\{0, 1, \dots, N - 1\}$ . Note that  $e^{-\frac{j2\pi ft}{N}} = \cos\left(\frac{2\pi ft}{N}\right) - j \sin\left(\frac{2\pi ft}{N}\right)$ . Thus, the time signal  $x(t)$  and its frequency representation  $X(f)$  have the following discrete Fourier transform relationship:

$$X(f) = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} x(t) e^{-\frac{j2\pi ft}{N}}$$

$$x(t) = \frac{1}{\sqrt{N}} \sum_{f=0}^{N-1} X(f) e^{\frac{j2\pi ft}{N}}$$

50. Thus, the Discrete Fourier Transform (DFT) and its inverse the Inverse Discrete Fourier Transform (IDFT) are mathematical tools to convert between the time domain and frequency domain representations. Computationally efficient implementations of the DFT and IDFT are referred to as Fast Fourier Transforms (FFTs) and Inverse Fast Fourier Transforms (IFFTs). The constellation points, one per subcarrier, that are to be transmitted, are the frequency domain representation because the constellation points are the weights that multiply each sinusoid. An IDFT or IFFT converts this frequency domain representation to the corresponding time domain representation which then is processed by a digital-to-analog converter and an amplifier to be transmitted on the channel. At the receiver, the analog signal is processed by an analog-to-digital converter and an FFT or DFT to be converted to the frequency domain so the noisy constellation points can be recovered and then processed, perhaps by a Reed-Solomon decoder.

51. A distortion phenomenon called inter-symbol interference (ISI) causes multiple copies of transmitted symbols to arrive at the receiver at different times. This causes the received signal to include components of multiple symbols added together. ISI occurs in both wireless and wired communication systems and in both point-to-point and multi-cast or broadcast applications. ISI can be mitigated by equalization in the time domain, which seeks to recover the original transmitted symbols from the received combinations.



52. ISI may also be understood through its effects at different frequencies; in the frequency domain ISI induces a frequency-selective gain. Multicarrier transmission overcomes ISI by transmitting on multiple parallel sub-carriers with relatively narrow effective bandwidth. Each sub-carrier typically experiences a different gain because of frequency selectivity, but each sub-carrier experiences negligible or significantly reduced inter-symbol interference because the sub-carriers are narrowband relative to the overall bandwidth.

53. Multicarrier transmission was a standard tool in communications at the time of the alleged inventions. The Kineplex modem developed by the Collins Radio Company employed multicarrier transmission no later than 1957. As described on page 657 of the 1957 IRE journal paper “Binary Data Transmission Techniques for Linear Systems,” by Doelz, Heald, and Martin, “Data is transmitted by means of multiple tones in the voice band...” and specifically “[t]wenty tones, at different frequencies, are multiplexed to provide the 3000 bit per second capacity.”

54. Multicarrier transmission was studied in a variety of research publications in the 1960s including the following sampling of articles: The technical report “Digital communication over fixed time-continuous channels with memory, with special applications to telephone channels,” by Holsinger (1964) laid out the theoretical framework for multicarrier transmission as a capacity-achieving approach. The journal paper “The AN//GSVC-10 (KATHRYN) Variable Rate Data Modem for HF Radio” by Zimmerman and Kirsch (1967) described an implemented multicarrier system. The journal paper “Performance of an efficient, parallel data transmission system,” by Salzberg (1967), and his related U. S. Patent No. 3,511,936 entitled “Multiply Orthogonal System for Transmitting Data Signals Through Frequency Overlapping Channels,” filed in May 1967, described a multicarrier transmission system. The journal paper “A theoretical study of performance of an orthogonal multiplexing data transmission scheme,” by Chang and

Gibby (1968) presents a theoretical analysis of multicarrier transmission that investigates practical problems such as sampling time error, carrier phase offset, and non-ideal phase characteristics.

55. The idea of implementing multicarrier transmission using the discrete Fourier Transform (DFT) was understood in the literature by 1969 at the latest. Salz and Weinstein presented this idea in their 1969 conference paper at ACM entitled “Fourier Transform Communication System.” Subsequently, Weinstein and Ebert elaborated on the scheme in the 1971 *IEEE Transactions on Communication Technology* journal paper “Data Transmission by Frequency-Division Multiplexing Using the Discrete Fourier Transform.”

56. The benefit of converting an ISI channel into parallel, independent ISI-free channels using the DFT applies both to point-to-point scenarios and to multi-cast or broadcast scenarios. This would have been well understood by a POSA long before the time of the alleged inventions. Although the fundamental transmitter and receiver functionality is very similar, these two use cases have different acronyms. The use of DFT-based multicarrier transmission for point-to-point transmission is sometimes called discrete multi-tone (DMT). The use of DFT-based multicarrier transmission for broadcast or multicast transmission is sometimes called Orthogonal Frequency-Division Multiplexing (OFDM).

57. In the years between 1987 and 1991 papers were written describing both OFDM and DMT systems. In 1987, Alard and Lasalle described an OFDM system in their EBU paper “Principles of modulation and channel coding for digital broadcasting for mobile receivers.” In 1989-1991 John Cioffi and his students at Stanford University published a series of papers described applying DFT-based multicarrier transmission to the point-to-point transmission problem of transmitting data between a central office and a home over telephone lines called digital subscriber lines (DSLs). These include the conference papers “Crosstalk-limited performance of a

computationally efficient multichannel transceiver for high rate digital subscriber lines” by Tu, Chow, Dudevoir, and Cioffi, presented at the 1989 IEEE Global Telecommunications Conference, and “A computationally efficient adaptive transceiver for high-speed digital subscriber lines” by Chow, Tu and Cioffi and presented at the 1990 IEEE International Conference on Communications.

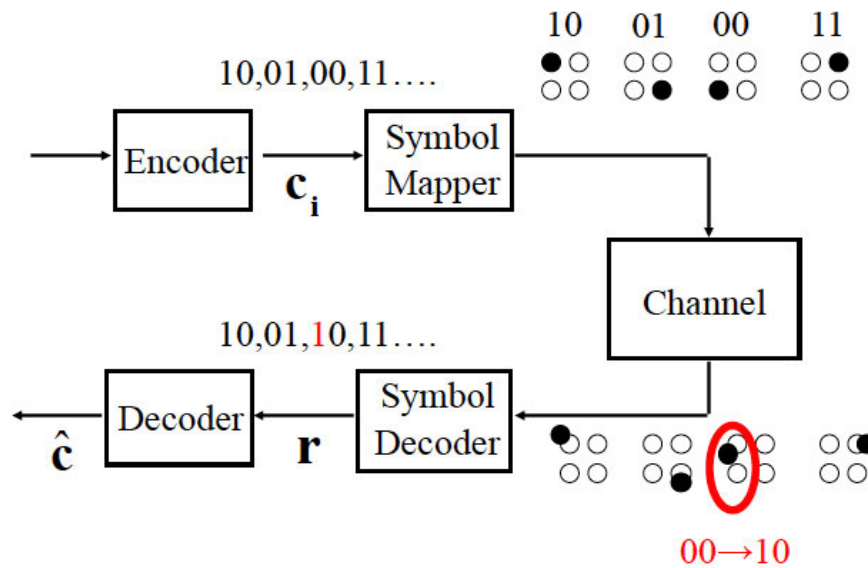
58. The journal paper “Performance evaluation of a multichannel transceiver system for ADSL and VHDSL services” by Chow, Tu, and Cioffi in the *IEEE Journal on Selected Areas in Communications* (JSAC) in 1991 and “A Multicarrier Primer,” by John Cioffi in 1991 described the DMT approach for DSLs which eventually evolved into the ADSL and VDSL standards. The Chow et al. JSAC article reiterates the basic advantage of the multicarrier approach on page 913: “The fundamental concept of multicarrier modulation is the conversion of a data transmission channel with inter-symbol interference (ISI), and possibly crosstalk and/or colored noise, into a set of parallel, independent and ISI-free subchannels.”

59. Additional papers on DMT and OFDM published in 1995 will be of interest in the discussions below. In 1995, Sari, Karam, and Jeanclaude explored OFDM for television broadcast in “Transmission techniques for digital terrestrial TV broadcasting,” in the *IEEE Communications Magazine*. Also in 1995 I wrote a paper along with Cioffi entitled “Fundamentals of coding for broadcast OFDM” at the 1995 Asilomar conference which explored coding and interleaving (two topics discussed below) for OFDM. For DMT systems Zogakis and Aslanis published “A Coded and Shaped Discrete Multitone System” in the *IEEE Transactions on Communications* in December of 1995, which presents a DMT system that employs Reed-Solomon coding and interleaving.

#### **B. Reed-Solomon Coding**

60. Bits are often communicated over channels as the labels of constellation points in Euclidean space (e.g., the coordinates  $x = 1$ ,  $y = 1$ ) using single-carrier or multicarrier modulation. When such bits are transmitted over noisy channels, the constellation points can be distorted

introducing bit errors as shown below in Figure 8. Error control coding, sometimes called forward error correction (FEC) coding, uses extra bits or symbols (sometimes called redundancy) to correct errors that may occur during transmission. The addition of the extra redundancy symbols can lower the probability of a message an error.



**Figure 8: An error control encoder followed by a symbol mapper that maps bits to constellation points. Noise in the channel distorts the constellation points leading to possible bit errors.**

61. A simple example of how this works is the three-bit repetition code. Consider a bit-error channel (often called the binary symmetric channel) that has a bit error probability of  $10^{-2}$  (or one in one hundred). If the transmitter sends a one-bit message on this channel, the receiver will observe an incorrect message with a probability of  $10^{-2}$  (that is, about one out of every hundred one-bit messages will be observed incorrectly). Suppose that instead of sending a single bit, the transmitter creates a codeword by repeating the message bit three times. When the receiver observes the three bits of the codeword, the receiver can select the message corresponding to the majority vote of the three observed bits. The extra two bits of redundancy reduce the probability

of a message error significantly, from 0.01 to about 0.0003 (a 1% chance versus an ~.03% chance of an error).

62. As discussed above, error control coding protects a message by using additional bits or symbols called redundancy to produce a codeword that can withstand errors in the channel. Reed-Solomon codes are a particular type of error control coding. The particular Reed-Solomon codes discussed in the patents at issue use 8-bit symbols, which are often called bytes or octets. A Reed Solomon code adds  $R$  extra symbols, sometimes called the  $R$  parity bytes or check bytes, to the  $K$  information symbols. The parameter  $R$  is referred to as the redundancy.

63. Reed-Solomon codes are a particularly powerful form of error correcting code because they achieve the Singleton bound. The Singleton bound involves Hamming distance. The Hamming distance between two codewords is simply the number of symbol positions where the codewords have different symbols. The Singleton bound says that for a code where each codeword has  $r$  symbols of redundancy, the minimum Hamming distance between codewords cannot be more than  $r + 1$ . Our simple example of a 3-bit repetition code achieves the Singleton bound by achieving a Hamming distance of 3 with redundancy  $r = 2$ . All Reed-Solomon codes achieve the Singleton bound, as stated and proved on page 175 of the classic 1983 textbook *Theory and practice of Error Control Codes* by Richard E. Blahut. That is, every Reed-Solomon code achieves the Hamming distance of at least  $r + 1$  between any two of its codewords.

64. Codes that achieve the Singleton bound, including all Reed-Solomon Codes, have the property that if the code has redundancy  $R$ , then the code will be able to correct any error pattern with  $\left\lfloor \frac{R}{2} \right\rfloor$  symbols. For example, a Reed-Solomon code with a redundancy of 16 parity bytes can correct any error pattern with 8 or fewer byte errors. An example of a Reed-Solomon code with 16

parity bytes is the (240,224) Reed-Solomon code with 224 information bytes and 16 parity bytes for a total of 240 bytes in each transmitted codeword.

65. Reed-Solomon coding had long been a tool commonly used by a POSA at the time of the alleged invention. Reed-Solomon codes were discovered by Reed and Solomon in 1960 and are a special case of non-binary BCH codes that were discovered independently by Bose and Ray-Chaudhuri in 1960 and Hocquenghem in 1959.

66. Error control coding in general, and Reed-Solomon coding in particular, was a standard tool in communications at the time of the alleged invention. For example, in the 1967 article describing the AN/GSC-10 KATHRYN multicarrier transmission system, the authors state that “[t]he AN/GSC-10 is compatible with error-correcting coding” without specifying the specific error control code. *The AN/GSC-10 (KATHRYN) Variable Rate Data Modem for HF Radio* at p. 198. Reed-Solomon codes would have been one choice that a POSA would have considered even in 1967.

67. As noted in Blahut’s 1983 textbook on p. 206:

**206 BOSE-CHAUDHURI-HOCQUENGHEM CODES**

**NOTES**

The BCH codes were discovered independently by Hocquenghem (1959) and by Bose and Ray-Chaudhuri (1960). It was noticed quickly by Gorenstein and Zierler (1961) that the codes discovered earlier by Reed and Solomon (1960) are a special case of nonbinary BCH codes.

68. The Gorenstein and Zierler 1961 article describes a decoder for BCH codes generally, and they also note that the codes discovered earlier by Reed and Solomon (1960) are a special case of non-binary BCH codes. An important improvement to the decoding procedure for BCH codes was identified for example by Berlekamp (1968) and re-interpreted by Massey (1969).

69. Thus Reed-Solomon codes were invented no later than 1960 (See I. S. Reed and G. Solomon, Polynomial Codes over certain Finite Fields, *J. Soc. Indust. Appl. Math*, 8, (1960): 300-304.) Well before the alleged invention of the asserted patents, Reed-Solomon codes were in common use. Blahut referred to them in his 1983 textbook as “An important and popular subset of the set of BCH codes.” See Blahut 1983 textbook at p. 174.

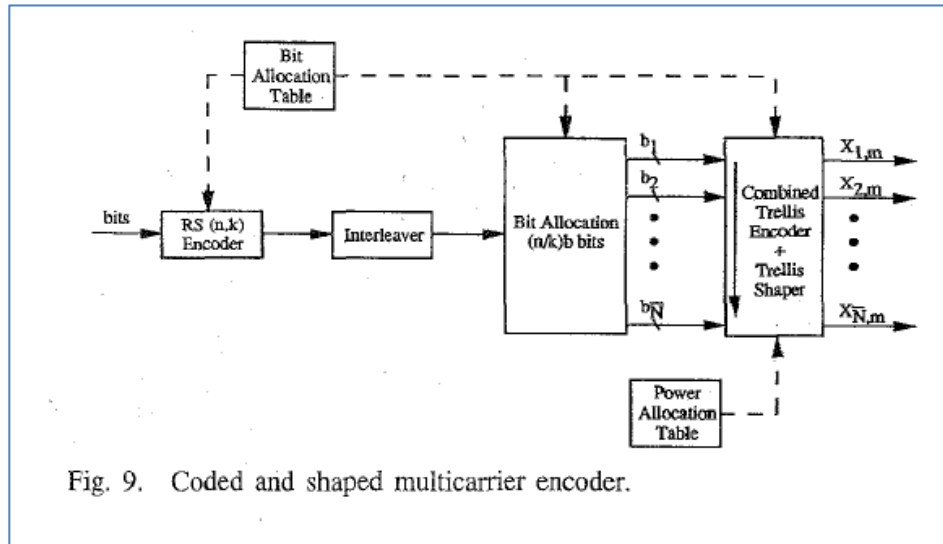
70. Reed-Solomon coding is a general purpose error control technique that can protect information in a wide variety of applications. Because they achieve the Singleton bound and had well-developed algebraic decoding algorithms, Reed-Solomon codes had already found many applications, including multicarrier transmission, by the time of the alleged invention.

71. As pointed out by Stephen B. Wicker in his 1995 book *Error Control Systems for Digital Communication and Storage*, (see page 14), “The non-binary Reed-Solomon codes are, quite simply, ubiquitous. A shortened pair of cross-interleaved Reed-Solomon codes provides error control for the compact digital audio disc (CD). Another Reed-Solomon code was used for error control during the Voyager exploration of the outer solar system.” Dr. Wicker goes on to say that “Reed-Solomon codes are easily tailored to meet the needs of a particular application.”

72. In the context of the state of the art as understood by a POSA, the use of Reed-Solomon codes in various communication systems at the time of the alleged invention, including a multicarrier modulation system with interleaving, would be an obvious choice to consider. In those cases where it had not already been explicitly done, it would certainly be applying a known improvement to achieve predictable results.

73. The use of Reed-Solomon coding in multicarrier transmission was certainly well-known by 1995. For a DMT, an example is illustrated in Figure 9 from page 2948 of the *IEEE Transactions on Communications* article, “A Coded and Shaped Discrete Multitone System” by

Zogakis and Aslanis from December of 1995, which is shown in Figure 9 below. In such a system, transmitted packets would contain one or more Reed-Solomon codewords. Note that the Reed-Solomon bytes produced by the RS(n,k) Encoder are interleaved.



**Figure 9:** In this figure, which is Figure 9 from the December 1995 Zogakis journal paper, we see an example of a system that used Reed-Solomon coding, interleaving, and multicarrier transmission.

74. The use of Reed-Solomon coding in an OFDM, i.e. broadcast or multicast multicarrier, scenario is illustrated in Figure 14 from page 180 of the EBU article, “Principles of modulation and channel coding for digital broadcasting for mobile receivers” by Alard and Lasalle from 1987, which is shown in Figure 10 below. In this figure, the “Block Code” is described on pages 179-180 as “an external Reed-Solomon code, as shown in the schematic diagram of Fig. 14.” The article considers both the standard Reed-Solomon code and cyclotomically shortened Reed-Solomon (CSRS) codes. In either case, transmitted packets would include one or more Reed-Solomon codewords.



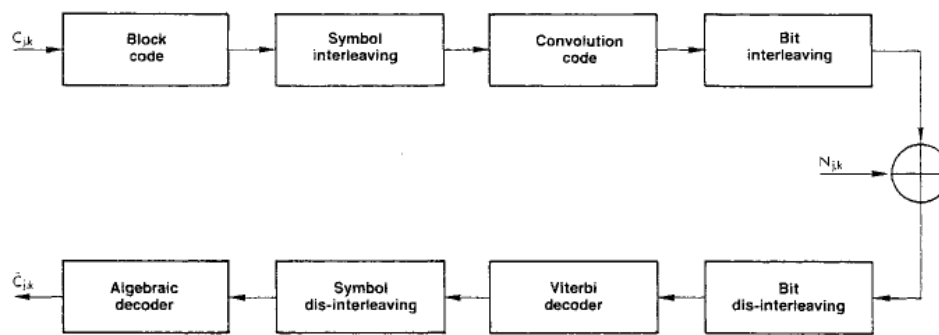


Figure 14  
Principle of concatenated codes

**Figure 10: In this figure, which is Figure 14 from the 1987 Alard and Lasalle journal paper, we see an example of a system that used Reed-Solomon coding, interleaving, and multicarrier transmission.**

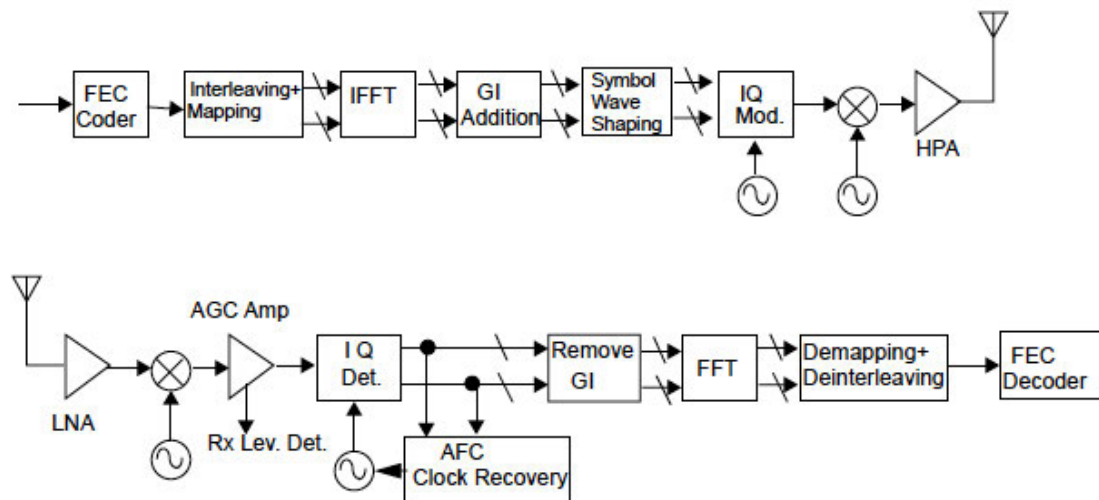
75. The use of Reed-Solomon coding in an Orthogonal Frequency-Division Multiplexing (OFDM), i.e. a broadcast or multicast, scenario was also considered in my 2000 journal paper with Liu and Shi entitled “Trellis Codes for Periodic Erasures,” where we designed new trellis codes for interleaved multicarrier transmissions and compared to a Reed-Solomon encoded, interleaved multicarrier transmission. For the Reed-Solomon encoded case, transmitted packets would include one or more Reed-Solomon codewords.

76. While Reed-Solomon coding was, and is, a common technique for error control coding used in a multicarrier transmission system, there are numerous alternatives for providing effective error control for multicarrier transmission. Popular alternatives to Reed Solomon codes include low-density parity-check (LDPC) codes, turbo codes, polar codes, convolutional codes, trellis codes, Reed-Muller codes, Golay codes, BCH codes that are not Reed-Solomon codes, and others. As discussed in the previous paragraph, I have personally designed trellis codes that, when utilizing soft Viterbi decoding, outperform Reed-Solomon codes in a multicarrier setting in certain scenarios.

77. Another example of alternative coding used with multi-carrier transmission is provided by the 1999 802.11a standard and specifically in the document *IEEE Standard 802.11a – 1999 (R2003), Part 11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, High-speed Physical Layer in the 5 GHz Band*. This document is available online at <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=815305>. Figure 11 below shows the general block diagram of the transmitter and receiver for 802.11a.

#### 17.3.8.1 Outline description

The general block diagram of the transmitter and receiver for the OFDM PHY is shown in Figure 118. Major specifications for the OFDM PHY are listed in Table 86.



**Figure 118—Transmitter and receiver block diagram for the OFDM PHY**

**Figure 11:** This figure, which is Figure 118 from the 1999 802.11a standard, provides an example of coding, interleaving, and multicarrier transmission where the forward error correction (FEC) code is a convolutional code rather than a Reed-Solomon code.

78. As we can see, 802.11a uses error control coding (the forward error correction (FEC) Coder transmitter module), interleaving (the Interleaving+Mapping transmitter module) and multicarrier transmission (the IFFT transmitter module). The error control coding used in 802.11a employs convolutional codes rather than a Reed-Solomon code. This is shown in Table 1 below.

**Table 86—Major parameters of the OFDM PHY**

<b>Information data rate</b>	6, 9, 12, 18, 24, 36, 48 and 54 Mbit/s (6, 12 and 24 Mbit/s are mandatory)
<b>Modulation</b>	BPSK OFDM QPSK OFDM 16-QAM OFDM 64-QAM OFDM
<b>Error correcting code</b>	K = 7 (64 states) convolutional code
<b>Coding rate</b>	1/2, 2/3, 3/4
<b>Number of subcarriers</b>	52
<b>OFDM symbol duration</b>	4.0 $\mu$ s
<b>Guard interval</b>	0.8 $\mu$ s <sup>2</sup> ( $T_{GI}$ )
<b>Occupied bandwidth</b>	16.6 MHz

<sup>2</sup>Refer to 17.3.2.4.

**Table 1: This table, which is Table 86 from the 1999 802.11a standard, provides information about the data rates, modulations, error correction codes, coding rates, and other parameters of the 802.11a physical layer.**

79. Note that three convolutional code rates are possible, 1/2, 2/3, and 3/4. Also, note that four modulations are possible, BPSK, QPSK, 16-QAM, and 64-QAM. A configuration message is required so that the transmitter can communicate to the receiver which convolutional code rate and which modulation are being used for a particular transmission, as will be discussed further below. It is a standard part of the design of a communications system to select the error control coding after evaluating the channel

### **C. Interleaving**

80. Interleaving re-orders (permutes) symbols so that impairments, such as a burst of noise, that impact consecutive transmitted symbols will be spread out over multiple codewords. Each Reed-Solomon codeword has a limit on the number of byte errors that it can correct. With interleaving, a relatively long burst of errors can be spread over multiple Reed-Solomon codewords, allowing all the errors to be corrected.

81. Interleaving was a standard tool in communications at the time of the alleged invention. In his 1971 journal paper in the *IEEE Transactions on Communications*, entitled “Burst-Correcting Codes for the Classic Bursty Channel,” G. David Forney observes on page 772 that:

The use of interleaving to adapt random-error correcting codes to bursty channels is frequently proposed ...

82. Two popular types of interleaving are block interleaving and convolutional interleaving. In 1971, Forney discussed both block and convolutional interleaving under the heading of “Archetypal Coding Schemes” on page 775 stating:

The usual type of interleaver is a block interleaver, in which, for example, bits are laid down in the rows of a  $B \times N$  matrix, and read out from the columns. In this paper we shall use a somewhat simpler and more effective type of interleaver, which we have called [11] a periodic (or convolutional) interleaver. (Similar interleavers were independently proposed by Cowell and Burton [12] and Ramsey [13])

83. In the quotation above, reference [12] is the 1962 paper “Computer Simulation of the use of Group Codes with Retransmission on a Gilbert Burst Channel” by W. R. Cowell and H. O. Burton published in the journal *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*.

84. On the first page of the Cowell and Burton article (page 577), they briefly describe the principle of using interleaving in conjunction with error control coding:

**Interleaving**

If errors occur in bursts, codes may be designed that are particularly suited to detecting or correcting this type of error pattern or the code words may be transmitted in an interleaved fashion so that digits of a code word are separated by digits of other code words. Thus, the effect of a burst is spread over several words and is less likely to exceed the error detecting or correcting capabilities of any one word.

85. Reference [13] in the quotation above from page 775 of Forney is the 1970 paper by J. Ramsey entitled “Realization of Optimum Interleavers,” which appeared in the *IEEE Transactions on Information Theory*. Ramsey gives an excellent general description of an interleaver in the first sentence of his article as follows: “An interleaver is a device that rearranges the ordering of a sequence of symbols in some one-to-one deterministic manner.” Ramsey went on to describe how interleavers are often used in communication systems:

In many of the applications to communication technology, interleaving is used as an adjunct to coding for error correction. One technique, which is useful for some types of burst-error channels, is to insert an interleaver between the channel encoder and the channel. The interleaver redistributes the channel symbols so that the symbols from a codeword are mutually separated by somewhat more than the length of a “typical” burst of errors. Thus, interleaving effectively makes the channel appear like a random-error channel to the decoder. For some HF channels, this technique can improve the performance by one to three orders of magnitude [1].

86. The combination of multicarrier transmission, Reed-Solomon coding, and interleaving was well-known by 1995. It was used, for example in the multicarrier transmission system described in the *IEEE Transactions on Communications* article, “A Coded and Shaped Discrete Multitone System” by Zogakis and Aslanis published in December of 1995. Figure 9 of that article (See Figure 9 above.) shows an interleaver processing the Reed-Solomon encoded bytes before they are modulated using multicarrier modulation.

87. The use of interleaving in Orthogonal Frequency-Division Multiplexing (OFDM), i.e. a broadcast or multicast multicarrier scenario, is illustrated, for example, in Figure 14 (See Figure 10 above.) from page 180 of the EBU article, “Principles of modulation and channel coding for digital broadcasting for mobile receivers,” by Alard and Lasalle from 1987. In this figure interleaving and de-interleaving of Reed-Solomon symbols are clearly indicated.

88. The discussion, shown below, on page 178 of the EBU article in the section 5.2 entitled “Use of an interleaving arrangement” describes both “frequency interleaving,” and “time interleaving” and notes that frequency interleaving, i.e. the interleaving of multicarrier symbols, is “essential for fixed reception or “stationary mobile.”

- interleaving in time, which is very useful for mobile reception, even at slow speed. This interleaving can be very deep (several hundreds of milliseconds), because the constraints at the level of processing delay in the receiver are much less severe in digital sound broadcasting than in radiotelephony;
- frequency interleaving, which is essential for fixed reception or “stationary mobile”. Note that with OFDM the number of carriers per programme can be very large. In fact, the real independence of these carriers depends in particular on the product of the bandwidth used  $W$  and the standard deviation of the distribution of the delays. From this point of view, the existence of multiple paths can be considered as an advantage.

89. There is considerable discussion in the literature specifically describing interleaving in conjunction with multicarrier transmission and coding such as Reed-Solomon coding as an option for OFDM. In the Alard and Lasalle 1987 article, Alard and Lasalle state on page 178 that interleaving allows Reed-Solomon codes to simultaneously exploit the “two essential particularities” of the Rayleigh channel they are studying, “these two properties to be exploited simultaneously by associating an interleaving system matched to the length of the code symbols and a system for deleting the affected symbols in the case of a deep fade.” Ultimately, the article suggests using interleaved Reed-Solomon codes used in conjunction with convolutional codes.

90. The article by Sari et al. in 1995 states in its conclusions on page 108, “With coding, interleaving, and weighted decoding, OFDM signaling eventually surpasses the performance of single-carrier transmission...” While pointing out that OFDM has strong sensitivity to nonlinear

distortion and carrier synchronization difficulties, the Sari et al. article concludes that OFDM with coding and interleaving was a strong potential technique for digital terrestrial TV broadcasting. Indeed, from this article a POSA would conclude that OFDM is *only* a strong contender when used with coding and interleaving. My own paper at Asilomar studying OFDM in 1995 entitled “Fundamentals of coding for broadcast OFDM” focused on developing improved approaches for OFDM assuming that coding and interleaving would be used, never even considering OFDM without both coding and interleaving.

91. Thus, certainly by 1995 it was well understood that a communication system using multicarrier transmission would typically use coding and interleaving to perform well. I also pointed out the natural application of interleaving to multicarrier transmission in my 2000 journal paper with Liu and Shi entitled “Trellis Codes for Periodic Erasures,” where I began that paper with the following statement:

Partial-band interference often distorts frequency-hopped or multicarrier transmissions. This interference might be caused by a malicious jammer in a military communications scenario or by adjacent channel interference in a multicarrier digital audio broadcast scenario. In both cases, the partial-band interference is contiguous and may be transformed to a periodic erasure pattern by a block interleaver. This paper treats the design and analysis of trellis codes specifically for such periodic erasure scenarios.

92. While the focus of my 2000 paper was on trellis codes, the idea of using Reed-Solomon codes in conjunction with interleaving and multicarrier transmission was so well-fixed at that time that I used that configuration as a baseline for comparison. As I stated in the introduction, “Section II also includes a motivational comparison between a trellis code designed according to this paper and a Reed–Solomon (RS) code of comparable rate.” Thus, several years prior to the alleged inventions, the idea of using Reed-Solomon codes, interleaving, and multicarrier

transmission together in a system was considered a standard approach. As we have seen, several such systems were discussed in the literature.

93. As explained by Forney in the quote above in ¶82, block interleaving and convolutional interleaving are two popular types of interleaving. Each type of interleaving is defined by how it reorders the symbols that it processes. There are typically multiple implementations, with each implementation requiring a different implementation-specific amount of memory, which achieve the same reordering. As long as the reordering is the same, these implementations are understood by a POSA to implement the same interleaver.

94. One implementation that achieves the re-ordering of block interleaving is to write the symbols row-by-row into a  $B \times N$  matrix (a matrix with  $B$  rows and  $N$  columns) and read the symbols column-by-column from the matrix. For this typical implementation the interleaver and deinterleaver each require enough memory to store the entire  $B \times N$  matrix of symbols, that is,  $BN$  symbols of memory for the interleaver and  $BN$  symbols of memory for the deinterleaver.

95. As an example, consider using a block interleaver implemented using a matrix with  $B = 3$  rows and  $N = 7$  columns to reorder the input sequence shown below:

$$x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}, x_{20}.$$

96. Using the implementation discussed above, the input sequence is written row-by-row into a  $3 \times 7$  matrix as shown below:

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} \\ x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} & x_{20} \end{bmatrix}.$$

97. To produce the interleaved sequence, the symbols are read out of the above matrix column-by-column, producing the output sequence shown below:

$$x_0, x_7, x_{14}, x_1, x_8, x_{15}, x_2, x_9, x_{16}, x_3, x_{10}, x_{17}, x_4, x_{11}, x_{18}, x_5, x_{12}, x_{19}, x_6, x_{13}, x_{20}.$$



98. The implementation described above uses 21 memory locations, each storing one symbol, to implement this interleaver. Any implementation that produces this same reordering is understood by a POSA to implement this block interleaver. One implementation of the deinterleaver is to write the received sequence of interleaved symbols column-by-column into a  $3 \times 7$  matrix. This produces the same matrix as shown above in ¶96, so reading the symbols from this matrix row-by-row recovers the symbols in their original order.

99. Now consider convolutional interleaving. The two key parameters that specify the permutation or re-ordering that defines a convolutional interleaver are the interleaver block length  $I$  and the interleaver depth (or delay increment)  $D$ . For each group of  $I$  symbols with indices (modulo  $I$ ) of  $j = 0, 1, \dots, (I - 1)$ , symbol  $j$  experiences a delay of  $j \cdot (D - 1)$ . Consider the example of convolutional interleaving with  $I = 7$  and  $D = 4$ . The input sequence

$x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}$

is reordered so that the symbols appear as follows in the output sequence:

$x_0$	■	■	■	$x_1$	■	■	$x_7$	$x_2$	■	■	$x_8$	$x_3$	■	$x_{14}$	$x_9$	$x_4$	■	$x_{15}$	$x_{10}$	$x_5$	$x_{21}$	$x_{16}$	$x_{11}$	$x_6$	$x_{22}$	$x_{17}$	$x_{12}$	■	$x_{23}$	$x_{18}$	$x_{13}$	■	$x_{24}$	$x_{19}$	■	■	$x_{25}$	$x_{20}$	■	■	$x_{26}$	■	■	■	$x_{27}$
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45

100. In the example output sequence above, the position of each symbol in the output sequence after interleaving is shown below the symbol, and the ■ symbols indicate previous or subsequent symbol inputs to the convolutional interleaver, which are not part of our 28-symbol input sequence of interest. The presence of these other input symbols interspersed in the contiguous symbols of our input sequence is a difference between convolutional interleaving and block interleaving; with block interleaving, previous or subsequent input symbols are not present in the contiguous sequence of output symbols corresponding to one input block.

101. As another illustration with  $I = 7$  and  $D = 4$ , if the convolutional interleaver input was

The\_quick\_brown\_fox\_jumps\_up

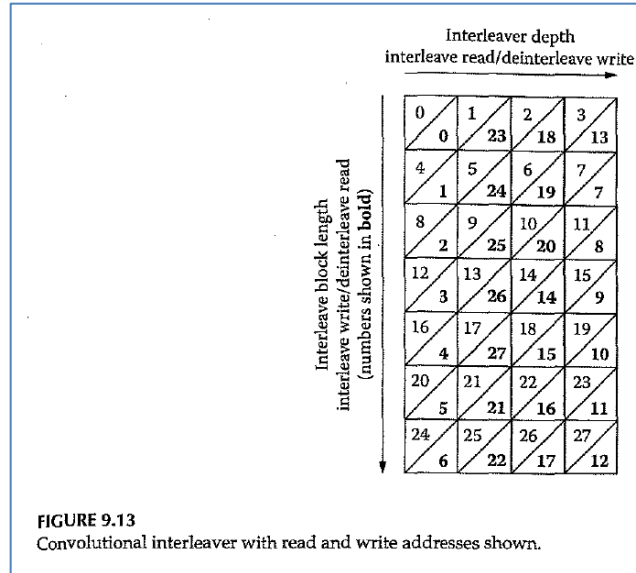
The convolutional interleaver output would be as follows:

T ■■■■ h■■■ c■■■ k\_ ■ n\_ q ■ \_ b u f r i m o o ■ p x w ■ s \_ ■■ \_ j ■■ u ■■■■ p

102. Again, the ■ symbols indicate previous or subsequent symbol inputs to the convolutional interleaver, which are not part of our 28-symbol input sequence of interest. Each group of letters is an interleaver block of length seven since  $I=7$ . For each interleaver block, adjacent letters are separated by  $D-1$  symbols. Looking at the interleaver output, note that, for example, each red letter is separated by  $D-1 = 3$  characters that are not red letters. The same is true for the blue letters, the black letters, and the brown letters.

103. Any implementation that accomplishes the reordering specified  $I$  and  $D$  is understood by a POSA to implement the corresponding convolutional interleaving. There are multiple implementations, with each implementation requiring a different implementation-specific amount of memory, which achieve the same reordering. As long as the reordering is the same, these implementations are understood by a POSA to implement the same convolutional interleaver.

104. *Fundamentals of DSL Technology*, which was published in 2006 and is not a prior art reference, describes how the reordering for convolutional interleaving specified by the parameters  $I$  and  $D$  can be implemented in a variety of ways using a variety of amounts of memory. For the same example considered above where  $I = 7$  and  $D = 4$ , Figure 9.13, shown below, from page 259 of *Fundamentals of DSL Technology* illustrates one example implementation, which uses 28 bytes of memory to store Reed-Solomon coded bytes, and possibly additional memory for pointers.



**Figure 12: A convolutional interleaver implementation that uses  $I \times D$  memory elements.**

105. Figure 12, which is Figure 9.13 from *Fundamentals of DSL Technology*, shows how a convolutional interleaver can be implemented using  $I \times D$  memory locations and possibly additional memory for pointers. In this example,  $I \times D = 28$ , and the figure shows the 28 memory locations for storing Reed-Solomon coded bytes and the order in which symbols are written into these memory locations for interleaving (the bold numbers shown in the bottom right of each square) as well as the order in which the symbols are read out of the memory locations to implement the specified re-ordering or permutation (the numbers in the upper left corner of each square). Here is the description from page 259 of *Fundamentals of DSL Technology*:

A convolutional interleaver is shown in Figure 9.13 for  $I = 7$  and  $d = 4$ . As with the block interleaver, symbols are written into the interleaver in columns and read out in rows. However, unlike the block interleaver, it is not necessary to wait to fill the entire block before reading. As shown in Figure 9.13, the first symbol, symbol 0, is written then read immediately with no delay. Symbol 1 is written, then read at time  $k = 4$  delayed by 3 samples. Symbol 2 is written then read out at time 8 delayed by 6 samples. For this example, the ordering of the input symbols in the output sequence is

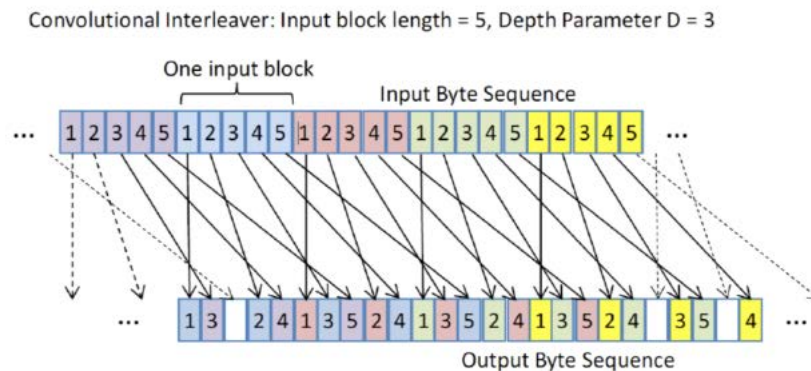
0, 4, 8, 12, 16, 20, 24, 7, 11, 15, 19, 23, 27, 31, 14, 18, ....

In general, for a convolutional interleaver, the ordering of the input symbols in the output sequence will be

$$0, d, 2d, \dots, (l-1) \cdot d, l, l+d, l+2d, \dots$$

106. Note that the ordering of the input symbols after convolutional interleaving described on page 259 of *Fundamentals of DSL Technology* matches the ordering shown in the example above in ¶99. Looking at the output sequence described in ¶96,  $x_0$  is in position 0,  $x_1$  is in position 4, and  $x_2$  is in position 8, and so on, as specified by the “ordering of the input symbols in the output sequence” in the excerpt from page 259 of *Fundamentals of DSL Technology*.

107. In his declaration for claim construction at ¶73 on page 24 Dr. Cooklev introduced a simple example of convolutional interleaving with  $I = 3$  and  $D = 5$ . Figure 13, below, shows an excerpt from ¶73 of Dr. Cooklev’s report on claim construction.



**Figure 13: Excerpt from Cooklev Claim Construction Declaration at ¶73 showing an example of convolutional interleaving with I=5 and D=3.**

108. The implementation described on page 259 of *Fundamentals of DSL Technology* can be used to implement the convolutional interleaving of the example at ¶73 of Dr. Cooklev’s claim construction report. The sequence of figures below shows how bytes are written into, and read out of, the fifteen-byte rectangle of memory locations using the same principles illustrated in Figure 9.13 from *Fundamentals of DSL Technology*. Each interleaver block of length I=5 is written

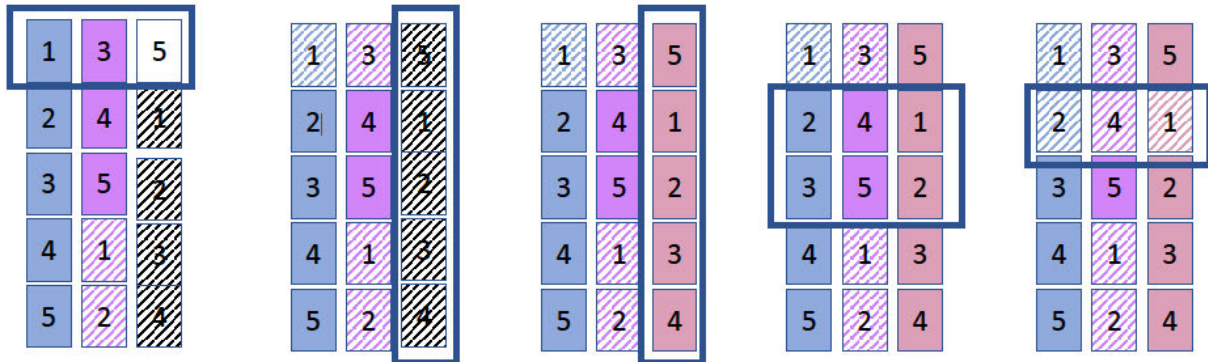
into a column of the rectangle, possibly after a permutation is applied. The interleaver output is read from the rectangle one row at a time.

109. Figure 14, shown below, includes five rectangles, each representing the same fifteen bytes of memory arranged in a matrix having three columns and five rows. Moving through the sequence of rectangles working left to right, we can see how writing a permuted version of each interleaver block into an available column and reading the output row by row produces the re-ordering of the convolutional interleaver of the example at ¶73 of Dr. Cooklev's claim construction report.

110. In the illustrations I have provided, each small rectangle with a number in it represents a byte. The colors and numbers match the colors and numbers of Dr. Cooklev's example except that I have added the appropriate numbers to the white squares that Dr. Cooklev left unnumbered. Diagonal stripes indicate that the byte has been read from the memory so that the location is now available to hold a new byte of interleaver input.

111. The leftmost rectangle of Figure 14 begins with the memory state just before the output sequence provide in Dr. Cooklev's example is produced. At that instant, the blue interleaver block has just been written into the leftmost column of memory, and that block (as are all blocks written in the leftmost column) is written in numerical order with no permutation. The purple interleaver block has been previously written into the middle column and was permuted (as are all blocks written in the middle column) so that the bytes were written into the column in the order 3,4,5,1,2. The output provided in Dr. Cooklev's example only includes the last three symbols from the purple interleaver block, i.e. the symbols numbered 3, 4, and 5. We can see that the purple bytes numbered 1 and 2 have diagonal stripes to indicate that they have already been read out of the memory. The rightmost column contains the white interleaver block of which only one symbol

(symbol 5, but unnumbered in Cooklev's illustration) has yet to be read and thus does not have diagonal stripes. These symbols were permuted (as are all blocks written in the middle column) so that the bytes were written into the column in the order 5,1,2,3,4.



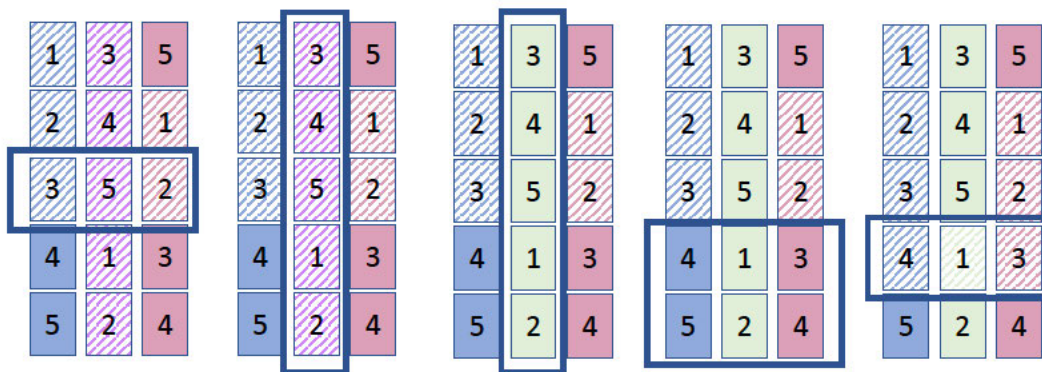
**Figure 14: Excerpt from page 1 of Exhibit 6 providing example of memory reads and writes for Dr. Cooklev's example of convolutional interleaving with  $I=5$  and  $D=3$  according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**

112. The interleaver can only read out a row that has three available bytes that have not been read, i.e., only rows with no diagonal stripes can be read. A blue rectangle highlights the only such row available in the leftmost rectangle of Figure 14. Reading out this row produces the first three bytes (the three leftmost bytes) of the output. A blue 1, a purple 3, and a white symbol, which is actually symbol 5. After reading out this row, we see that there are no available rows left to read.

113. The second (from the left) fifteen-byte rectangle in Figure 14 shows the state of the memory right after reading the top row of the previous rectangle. While there are no rows available to read, all of the bytes in the far-right column have diagonal stripes indicating that this column is available for writing. This is emphasized by the blue rectangle highlighting the far-right column.

114. The third rectangle of Figure 14 shows the next interleaver block (the pink block) being written into the available column in the permuted order 5,1,2,3,4 that is always used for the right column.

115. The blue rectangle in the fourth rectangle of Figure 14 indicates the two rows that are available for reading now that the pink interleaver block has been written into the memory. Reading out these two rows produces the next six symbols of output in Dr. Cooklev's example as follows: blue 2, purple 4, pink 1 (highlighted in the fifth rectangle of Figure 14) and blue 3, purple 5, pink 2 (highlighted in the first rectangle of Figure 15, shown below).



**Figure 15: Excerpt from page 2 of Exhibit 6 providing example of memory reads and writes for Dr. Cooklev's example of convolutional interleaving with I=5 and D=3 according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**

116. The second fifteen-byte rectangle of Figure 15 shows that the middle column of the memory is now available.

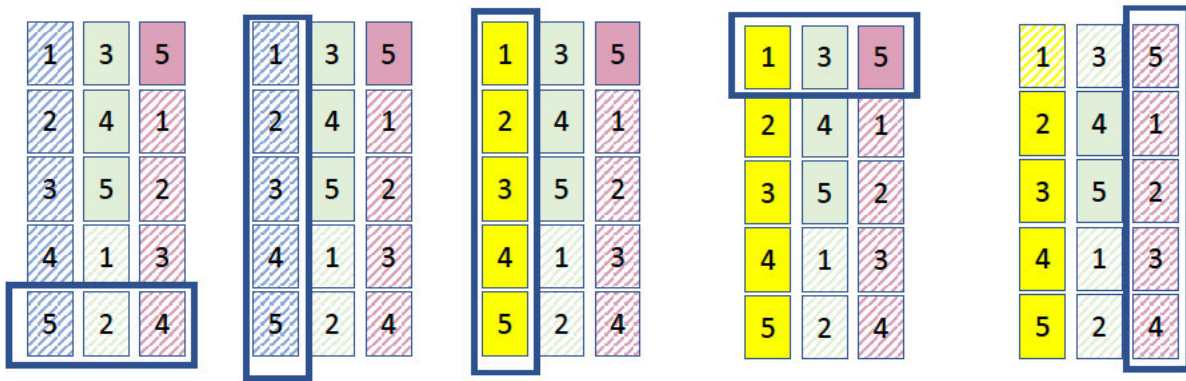
117. The third fifteen-byte rectangle of Figure 15 indicates that the next interleaver input block (the green block) is written into the available column using the permutation 3,4,5,1,2 that is always used for the middle column.

118. The fourth fifteen-byte rectangle of Figure 15 shows that after writing the green input interleaver block the bottom two rows are available for reading.



119. The fifth fifteen-byte rectangle of Figure 15 shows that reading the first of these two available rows produces the output blue 4, green 1, pink 3, the next three symbols of output in Dr. Cooklev's example.

120. The first fifteen-byte rectangle of Figure 16, shown below, indicates that reading the second of these two available rows produces the output blue 5, green 2, pink 4, which is the next three symbols of output in Dr. Cooklev's example.



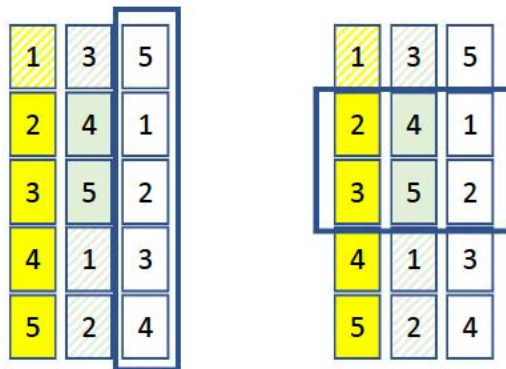
**Figure 16: Excerpt from page 3 of Exhibit 6 providing example of memory reads and writes for Dr. Cooklev's example of convolutional interleaving with  $I=5$  and  $D=3$  according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**

121. The second fifteen-byte rectangle of Figure 16 of shows that the first column (left column) of the memory is now available for writing. The third fifteen-byte rectangle of Figure 16 indicates that the next interleaver input block (the yellow block) is written into the available column using the permutation 1,2,3,4,5 (i.e. no permutation) that is always used for the left column. The fourth fifteen-byte rectangle of Figure 16 shows that after writing the yellow input interleaver block the top row is available for reading. Reading this row produces the output yellow 1, green 3, pink 5, which are the next three symbols of output in Dr. Cooklev's example. The sixth fifteen-byte



rectangle of Figure 16 of shows that after reading the top row, the third column (right column) of the memory is now available.

122. The first fifteen-byte rectangle of Figure 17, shown below, indicates that the next interleaver input block, a white block not explicitly shown in Dr. Cooklev's example, is written into the available column using the permutation 5,1,2,3,4 that is always used for the right column. The second fifteen-byte rectangle of Figure 17, the final such rectangle of this explanation, shows that the second and third rows are now available to read. Reading these two rows produces the next six symbols of Dr. Cooklev's example, yellow 2, green 4, white 1, yellow 3, green 5, white 2.



**Figure 17: Excerpt from page 4 of Exhibit 6 providing example of memory reads and writes for Dr. Cooklev's example of convolutional interleaving with I=5 and D=3 according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**

123. The figures described in ¶¶107-122 above were presented to Dr. Cooklev as Exhibit 6 of his deposition. Dr. Cooklev had not investigated possible implementations of his example convolutional interleaver the time of his deposition (Cooklev Claim Const. Dep. at 52:12). While he agreed with the individual steps of the implementation in Exhibit 6 of his deposition (Cooklev Claim Const. Dep. at 52--66) that I have described above, he was unable to confirm whether the structure implemented convolutional interleaving without further study (Cooklev Claim Const. Dep. at 67:2-9).

124. The Family 3 patent specifications describe embodiments in relation to DSL modems. See, e.g. ‘882 3:6-19, 6:65-7:3, 7:50-59, 10:39-50. At the time of the alleged invention convolutional interleaving was used, for example, in multiple DSL standards, including G.992.1, G.992.2, G.992.3, and G.993.1 (VDSL1). For G.992.1 and G.992.2, no specific implementation of the convolutional interleaver is described in the standard. However, as mentioned below, VDSL1 gives a detailed description of the triangular implementation constraints for the interleaver settings, for example constraining depth to have the form  $D = M \times I + 1$ , so that this implementation can always be used. Convolutional interleaving is described in G.992.2 as follows:

#### 7.6 Interleaver

The Reed-Solomon codewords shall be convolutionally interleaved. The interleaving depth shall always be a power of 2. Convolutional interleaving is defined by the rule:

Each of the  $N_{\text{FEC}}$  bytes  $B_0, B_1, \dots, B_{N-1}$  in a Reed-Solomon codeword is delayed by an amount that varies linearly with the byte index. More precisely, byte  $B_i$  (with index  $i$ ) is delayed by  $(D - 1) \times i$  bytes, where  $D$  is the interleave depth.

An example for  $N_{\text{FEC}} = 5$ ,  $D = 2$  is shown in Table 6, where  $B_i^j$  denotes the  $i$ -th byte of the  $j$ -th codeword.

18 **Recommendation G.992.2 (06/99)**

**Table 6/G.992.2 – Convolutional interleaving example for  $N_{\text{FEC}} = 5$ ,  $D = 2$**

Interleaver input	$B_0^j$	$B_1^j$	$B_2^j$	$B_3^j$	$B_4^j$	$B_0^{j+1}$	$B_1^{j+1}$	$B_2^{j+1}$	$B_3^{j+1}$	$B_4^{j+1}$
Interleaver output	$B_0^j$	$B_3^{j-1}$	$B_1^j$	$B_4^{j-1}$	$B_2^j$	$B_0^{j+1}$	$B_3^j$	$B_1^{j+1}$	$B_4^j$	$B_2^{j+1}$

With the above-defined rule, and the chosen interleaving depths (powers of 2), the output bytes from the interleaver always occupy distinct time slots when  $N_{\text{FEC}}$  is odd. When  $N_{\text{FEC}}$  is even, a dummy byte shall be added at the beginning of the codeword at the input to the interleaver. The resultant odd-length codeword is then convolutionally interleaved, and the dummy byte shall then be removed from the output of the interleaver.

(Recommendation G.992.2 (06/99) pages 18-19)

125. The VDSL1 description of Interleaving is provided, for example, in section 8.4, which is provided below:

#### 8.4 Interleaving

##### 8.4.1 General

Interleaving shall be used to protect the data against bursts of errors by spreading the errors over a number of Reed-Solomon codewords. The interleave depth shall be programmable with a maximum interleave depth of 64 codewords when the number of octets per codeword ( $N$ ) equals 255. For smaller values of  $N$  the interleave depth can grow nearly proportionately.

It shall be possible to adjust the interleave depth via the management system to meet latency requirements. The latency of the slow path is a function of the data rate and burst error correction capability. For data rates greater than or equal to 13 Mbit/s, the latency between the  $\alpha$  and  $\beta$  interfaces shall not exceed 10 ms when the interleaver depth is set to the maximum. At lower data rates there is a trade-off between higher latency and decreased burst error correction ability. At any data rate, the minimum latency occurs when the interleaver is turned off.

When the interleaver is on, the codewords shall be interleaved before transmission to increase the immunity of RS codewords to bursts of errors. The convolutional interleaver is defined by two parameters: the interleaver block length,  $I$ , and the interleaving depth,  $D$ . The block length  $I$  shall divide the RS codeword length  $N$  (i.e.,  $N$  shall be an integer multiple of  $I$ ). The convolutional interleaver uses a memory in which a block of  $I$  octets is written while an (interleaved) block of  $I$  octets is read. Details of the implementation are given in 8.4.2.

The same size interleaving memory (see Table 8-1) is needed for interleaving at the transmitter and de-interleaving at the receiver.

The convolutional interleaving introduces an absolute read-to-write delay,  $\Delta_j$ , that increments linearly with the octet index within a block of  $I$  octets:

$$\Delta_j = (D-1) \times j$$

where  $j = 0, 1, 2, \dots, I-1$ .

(VDSL1 § 8.4.1 “Interleaving – General”.)

126. As observed by Dr. Cooklev at ¶73 his Declaration for Claim Construction, “The DSL standards use a convolutional interleaver.” At his deposition on claim construction, Dr. Cooklev confirmed that ITU-T G.992.3 (ADSL2), ITU-T G.993.1 (VDSL1), and ITU-T G.993.2 (VDSL2) all specify convolutional interleaving. DSL standards to which the Family 3 patents are directed use convolutional interleaving, which can be implemented in a variety of ways including the implementation described above in ¶¶105-122.

127. The Family 3 patents assume an interleaver implementation that uses the amount of memory  $I \times D$  or  $N \times D$  computed as described in this example from *Fundamentals of DSL Technology*. For example, the Family 3 patents describe that an interleaver having a codeword size ( $N$ ) of 255 bytes and an interleaver depth ( $D$ ) of 64 will use 16 kbytes of memory ( $N \times D$  or  $255 \times 64 = 16320$  bytes). This can be seen in the ‘882 patent at 6:20-47 shown below. The ‘882 patent

similarly discloses computing the interleaver memory as  $N \times D$  in Examples #2 and #3, as shown below in the excerpts ‘882 7:5-25 and 7:29-49. As discussed above, one way to implement a convolutional interleaver is the implementation described above in ¶¶105-122, which uses  $I \times D$  or  $N \times D$  memory elements to store Reed-Solomon bytes. The Family 3 patents do not describe computing interleaver memory in a manner other than computing the interleaver memory as  $N \times D$ .

#### Example #1

20 A first transmitter portion or receiver portion latency path may carry data from a video application, which needs a very low BER but can tolerate higher latency. In this case, the video will be transported using a latency path that has a large amount of interleaving/deinterleaving and coding (also  
25 known as Forward Error Correction (FEC) coding). For example, the latency path may be configured with Reed-Solomon coding using a codeword size of 255 bytes ( $N=255$ ) with 16 checkbytes ( $R=16$ ) and interleaving/deinterleaving using an interleaver depth of 64 ( $D=64$ ). This latency path  
30 will require  $N \times D = 255 \times 64 = 16$  Kbytes of interleaver memory at the transmitter (or de-interleaver memory at the receiver). This latency path will be able to correct a burst of errors that is less than 512 bytes in duration.

35 A second transmitter portion or receiver portion latency path may carry an internet access application that requires a medium BER and a medium amount of latency. In this case, the internet access application will be transported using a latency path that has a medium amount of interleaving and coding. For example, the latency path may be configured with  
40 Reed-Solomon coding using a codeword size of 128 bytes ( $N=128$ ) with 8 checkbytes ( $R=8$ ) and interleaving using an interleaver depth of 16 ( $D=32$ ). This latency path will require  $N \times D = 128 \times 32 = 4$  Kbytes of interleaver memory and the same amount of deinterleaver memory. This latency path will be  
45 able to correct a burst of errors that is less than 128 bytes in duration.

‘882 6:20-47

## Example #2

5

If instead of 1 video application, 1 internet application and 1 voice application, there were 3 internet access applications then the transmitter portion and/or receiver portion latency paths would be reconfigured to utilize the shared memory and coding module in a different way. For example, the system could be reconfigured to have 3 transmitter portion or receiver portion latency paths, with each latency path being configured with Reed-Solomon coding using a codeword size of 128 bytes ( $N=128$ ) with 8 checkbytes ( $R=8$ ) and interleaving using an interleaver depth of 16 ( $D=32$ ). Each latency path will require  $N \times D = 128 \times 32 = 4$  Kbytes of interleaver memory and each block will be able to correct a burst of errors that is less than 128 bytes in duration. Based on the example of carrying the three internet access applications described, the three latency path share one memory space containing at least  $3 \times 4 = 12$  Kbytes. Also the three latency paths share a common coding block that is able to simultaneously encode (on the transmitter side) or decode (on the receiver side) three codewords with  $N=128/R=16$ ,  $N=128/R=8$  and  $N=128/R=8$ .

‘882 7:5-25

## Example #3

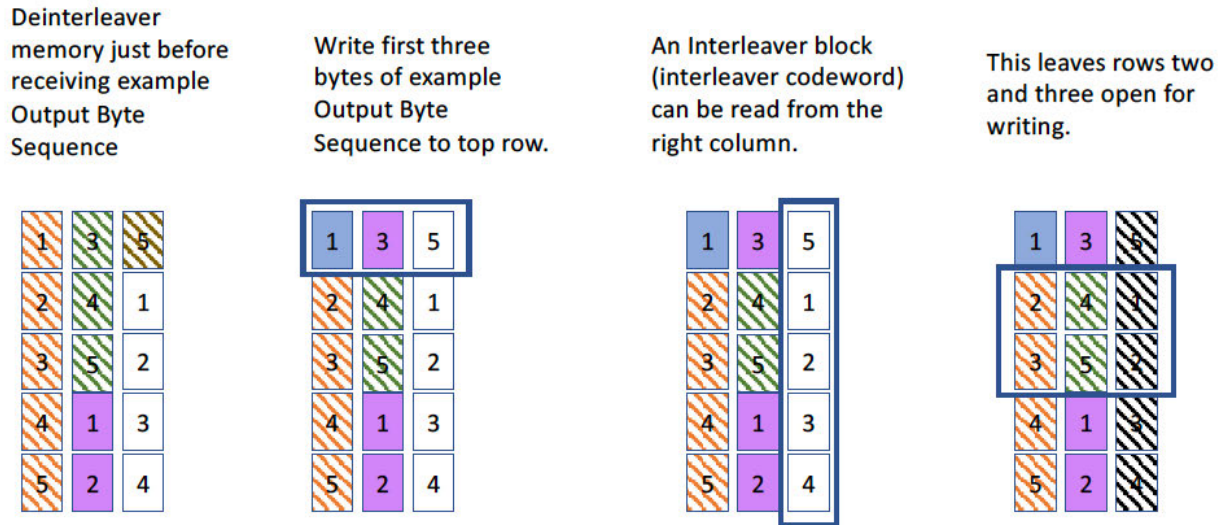
The system could be configured to carry yet another set of applications. For example, the latency paths could be configured to carry 2 video applications. In this case only 2 transmitter portion or receiver portion latency paths are needed, which means that the third latency path could be simply disabled. Also, assuming that the memory is constrained based on the first example above, then the maximum shared memory for these 2 latency paths is 20 kBytes. In this case, the system could be reconfigured to have 2 latency paths, with each block being configured with Reed-Solomon coding using a codeword size of 200 bytes ( $N=200$ ) with 10 checkbytes ( $R=10$ ) and interleaving/deinterleaving using an interleaver depth of 50 ( $D=50$ ). Each latency path will require  $N \times D = 200 \times 50 = 10$  Kbytes of interleaver memory and each block will be able to correct a burst of errors that is less than 250 bytes in duration. This configuration results in 20K of shared memory for both latency paths, which is the same as in the first example. In order to stay within the memory constraints of the latency paths, the error correction capability for each latency path is decreased to 250 bytes from 512 bytes in Example #1.

‘882 7:29-49

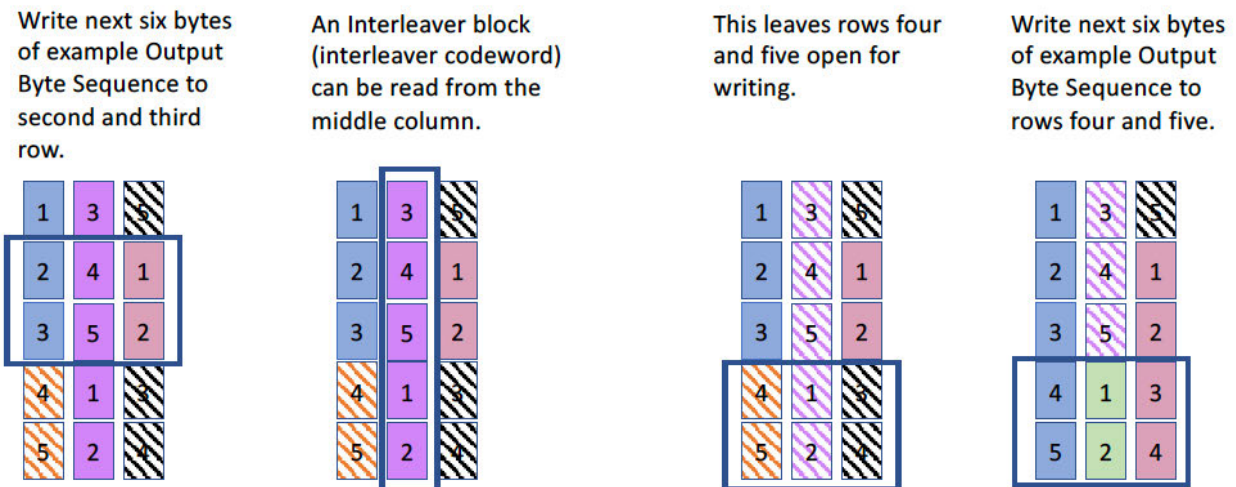
128. The interleaver implementation described above in ¶¶105-122 has a corresponding deinterleaver implementation that also uses  $I \times D$  memory locations and possibly additional memory for pointers. The figures below, Figure 18 through Figure 21, describe, in a similar manner



as above, how a fifteen-byte memory for storing Reed-Solomon coded bytes can be used (possibly in conjunction with additional memory for pointers) to implement a deinterleaver that correctly deinterleaves a received sequence. The figures below show writing the received sequence into the memory row by row and then reading out the deinterleaved sequence by reading columns when they are available using the appropriate permutation.



**Figure 18: Illustration of memory reads and writes for deinterleaving Dr. Cooklev's example of convolutional interleaving with  $I=5$  and  $D=3$  according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**



**Figure 19: Continued illustration of memory reads and writes for deinterleaving Dr. Cooklev's example of convolutional interleaving with  $I=5$  and  $D=3$  according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**

An Interleaver block (interleaver codeword) can be read from the left column.

1	3	5
2	4	1
3	5	2
4	1	3
5	2	4

This leaves row one open for writing.

1	3	5
2	4	1
3	5	2
4	1	3
5	2	4

Write next three bytes of example Output Byte Sequence to top row.

1	3	5
2	4	1
3	5	2
4	1	3
5	2	4

**Figure 20: Continued illustration of memory reads and writes for deinterleaving Dr. Cooklev's example of convolutional interleaving with  $I=5$  and  $D=3$  according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**

An Interleaver block (interleaver codeword) can be read from the right column.

1	3	5
2	4	1
3	5	2
4	1	3
5	2	4

This leaves rows two and three open for writing.

1	3	5
2	4	1
3	5	2
4	1	3
5	2	4

Write next six bytes of example Output Byte Sequence to rows two and three.

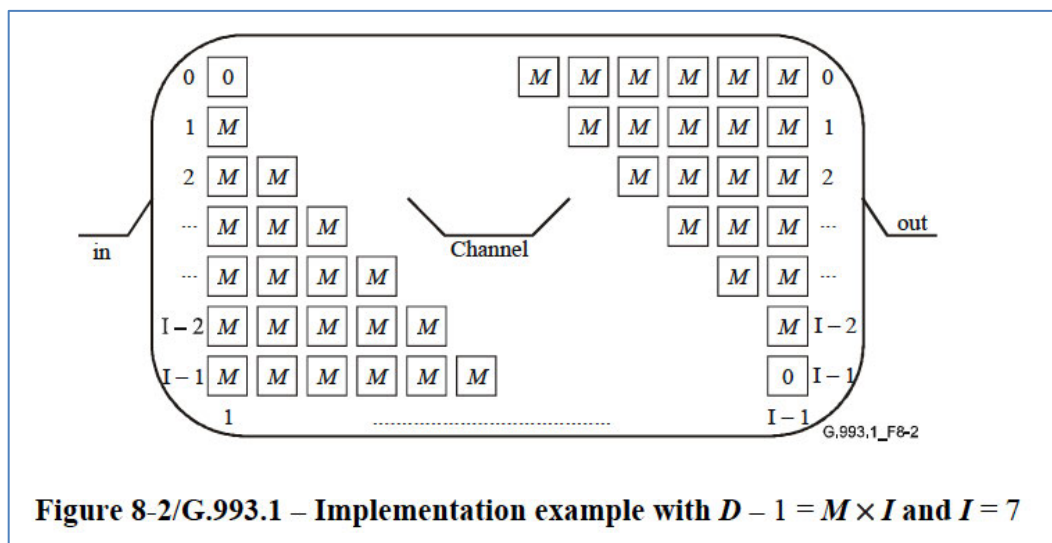
1	3	5
2	4	1
3	5	2
4	1	3
5	2	4

An Interleaver block (interleaver codeword) can be read from the middle column.

1	3	5
2	4	1
3	5	2
4	1	3
5	2	4

**Figure 21: Conclusion of illustration of memory reads and writes for deinterleaving Dr. Cooklev's example of convolutional interleaving with  $I=5$  and  $D=3$  according to the implementation technique exemplified in Figure 9.13 on page 259 of *Fundamentals of DSL Technology*.**

129. Another implementation of convolutional interleaving is the triangular implementation, shown in Figure 22 below, which is a reproduction of Figure 8-2 from the VDSL1 standard. The triangular implementation requires that the interleaver depth  $D$  is always chosen as one more than an integer multiple of the interleaver block length  $I$ , i.e.  $D = M \times I + 1$  or  $D-1 = M \times I$ , where  $M$  is some positive integer. Note that neither of the two simple examples discussed above where  $I=7$  and  $D=4$  and where  $I=5$  and  $D=3$  can meet this condition because in both cases  $D-1$  is smaller than  $I$  and thus there does not exist an integer  $M$  for which  $D-1 = I \times M$ .



**Figure 22: Implementation of triangular implementation for an interleaver with  $D-1 = M \times I$  and  $I=7$  as illustrated in Figure 8-2 of G.993.1.**

130. Figure 22 shows the triangular implementation of an interleaver with parameters  $I$  and  $M$ . Each box containing the letter  $M$  represents  $M$  memory elements used to delay bytes written into the interleaver. Figure B.2 shows  $I$  rows corresponding to first-in-first-out (FIFO) buffers implemented as delay lines or shift registers (or virtual shift registers (See VDSL1 at 152-153.)



implemented, for example, in a random access memory), with indices  $0, \dots, I-1$ . At the transmitting transceiver,  $I$  Reed-Solomon bytes at a time are written into the interleaver at the left side of each shift register. Then,  $I$  bytes are read from the interleaver at the right side of each shift register and are transmitted. Bytes written to the top shift register experience a delay of zero bytes before transmission. Bytes written to the next shift register experience a delay of  $M$  bytes. Bytes written to the shift register after that experience a delay of  $2M$  bytes and so on. The bottom shift register is delayed by  $(I-1) \times M$  bytes before transmission.

131. To undo the interleaving, the deinterleaver also uses  $I$  shift registers, with indices  $0, \dots, I-1$ , which are matched to the shift registers of the interleaver. The top shift register of the deinterleaver will delay its inputs by  $(I-1) \times M$  bytes, the next shift register delays inputs by  $(I-2) \times M$  bytes, and so on. In the deinterleaver, the bottom shift register implements a delay of zero bytes. In this way, the overall effect of the combination of the interleaver and deinterleaver is to delay each column by  $(I-1) \times M$  bytes. Considering that each of the  $I$  bytes in a column is transmitted separately, the overall delay experienced by each byte is exactly  $I \times (I-1) \times M$ . Thus, the deinterleaver returns the interleaved sequence back to its original order, but the combination of interleaving and deinterleaving delays the sequence by  $I \times (I-1) \times M$  bytes.

132. Figure 22 shows how the parameters  $I$  and  $M$  determine the (ideal) amount of memory that needs to be allocated for the triangular implementation of a convolutional interleaver. Each square containing the letter  $M$  represents  $M$  memory elements used to delay bytes written into the interleaver. Counting up the boxes, there are  $I-1$  in the first column,  $I-2$  in the second column, and so on until the last column, which has only one box containing  $M$  memory elements. Since the sum of  $(I-1) + (I-2) + (I-3) + \dots + 1 = \frac{I \times (I-1)}{2}$ , the total amount of memory that we need to allocate for the interleaver is precisely  $M \times \frac{I \times (I-1)}{2}$ . The same argument shows that the associated

deinterleaver with parameters  $I$  and  $M$  also requires an allocation of  $M \times \frac{I \times (I-1)}{2}$  bytes of memory for the specific triangular implementation.

133. Thus, for the triangular implementation, the equation  $M \times \frac{I \times (I-1)}{2} = (I-1)(D-1)/2$  gives the (ideal) amount of memory used by the triangular interleaver implementation shown in Figure 8-2. The expression  $M \times I \times (I-1)/2 = (I-1)(D-1)/2$  also gives the (ideal) amount memory used by the corresponding deinterleaver implementation. The ideal amount of memory calculated above does not include additional memory such as pointers that would be required to implement the interleaver or deinterleaver function using the common approach of implementing each first-in-first-out (FIFO) buffer as a circular buffer in random access memory (RAM). The ideal amount of memory also assumes that the bytes entering the top FIFO of the interleaver or the bottom FIFO of the deinterleaver are never written to the interleaver FIFO memory passing immediately from input to output. Actual implementations will commonly write all the bytes including those entering top FIFO of the interleaver or the bottom FIFO of the deinterleaver into the interleaver or deinterleaver memory.

134. Section 8.4.2 of VDSL1 gives a detailed description of the typical “triangular” implementation of a convolutional interleaver. This implementation shows how the interleaver/deinterleaver memory size for this typical implementation can be deduced from the two parameters  $M$  and  $I$  as  $M \times I \times (I - 1)/2$  bytes. In Figure 22 above, the interleaver memory depicted as the left triangle of blocks of  $M$ -byte memory elements. The deinterleaver memory is the same size and is depicted as the inverted triangle on the right, again constructed of  $M$ -byte memory elements.

#### 8.4.2 Triangular implementation

To decrease the implementation complexity, the delay increment ( $D - 1$ ) shall be chosen as a multiple of the interleaver block length ( $I$ ), i.e.:  $D - 1 = M \times I$ . The  $(D - 1)$  to  $I$  ratio is the interleaving depth parameter ( $M$ ). The characteristics of convolutional interleaving are shown in Table 8-1. The parameters  $t$  and  $q$  depend on the characteristics of the RS code and are defined as:

- $t$  = number of bytes that can be corrected by RS codewords = half the number of redundancy bytes =  $R/2$ ;
- $q$  = length of RS codeword divided by the length of an interleaver block =  $N/I$ .

**Table 8-1/G.993.1 – Characteristics of convolutional interleaving**

Parameter	Value
Interleaver block length ( $I$ )	$I$ bytes (equal to or divisor of $N$ )
Interleaving Depth ( $D$ )	$M \times I + 1$
(De)interleaver memory size	$M \times I \times (I - 1)/2$ bytes
Correction capability	$\lfloor t/q \rfloor \times (M \times I + 1)$ bytes
End-to-end delay	$M \times I \times (I - 1)$ bytes

The example in Figure 8-2 shows  $I = 7$ .  $I$  parallel branches (numbered  $0 \dots I - 1$ ) are implemented with a delay increment of  $M$  octets per branch. Each branch shall be a FIFO shift register (delay line) with length  $0 \times M \dots (I - 1) \times M$  bytes. The deinterleaver is similar to the interleaver, but the branch indices are reversed so that the largest interleaver delay corresponds to the smallest deinterleaver delay. Deinterleaver synchronization shall be achieved by routing the first byte of an interleaved block of  $I$  bytes into branch 0.

(VDSL1 § 8.4.2.)

135. Table 8-2 below shows the interleaver/deinterleaver memory sizes that correspond to various values of  $M$  and  $I$  using the equation  $M \times I \times (I - 1)/2$  bytes to compute the interleaver/deinterleaver memory sizes.

**Table 8-2/G.993.1 – Example of interleaver parameters with RS(144,128)**

Rate [kbit/s]	Interleaver parameters	Interleaver depth	(De)interleaver memory size	Erasure correction	End-to-end delay
$50 \times 1024$	$I = 72$ $M = 13$	937 blocks of 72 bytes	33 228 bytes	3 748 bytes 520 $\mu$ s	9.23 ms
$24 \times 1024$	$I = 36$ $M = 24$	865 blocks of 36 bytes	15 120 bytes	1 730 bytes 500 $\mu$ s	8.75 ms
$12 \times 1024$	$I = 36$ $M = 12$	433 blocks of 36 bytes	7 560 bytes	866 bytes 501 $\mu$ s	8.75 ms
$6 \times 1024$	$I = 18$ $M = 24$	433 blocks of 18 bytes	3 672 bytes	433 bytes 501 $\mu$ s	8.5 ms
$4 \times 1024$	$I = 18$ $M = 16$	289 blocks of 18 bytes	2 448 bytes	289 bytes 501 $\mu$ s	8.5 ms
$2 \times 1024$	$I = 18$ $M = 8$	145 blocks of 18 bytes	1 224 bytes	145 bytes 503 $\mu$ s	8.5 ms

(VDSL1 § 8.4.2.)

136. VDSL1 interleaving is further described in Annex I in Section I.5 “Complementary information on QAM Implementation” (informative), specifically in section I.5.4 Parameters of the Interleaver, which is included below. This description indicates the use of virtual shift registers, i.e. implementation of shift register behavior using a general purpose memory that is accessed using addresses. The length of each virtual shift register, i.e. the number of bytes of memory occupied by each virtual shift register, is described as  $M \times j$ . Summing the memory required by each virtual shift register produces the total memory requirement of  $M \times I \times (I - 1)/2$ .

137. To see how any convolutional interleaver (regardless of implementation) spreads a noise burst so that it can be successfully decoded by Reed-Solomon codewords, consider the following simple example. Let’s suppose our codewords have length  $N = 8$  and that each codeword can correct two errored bytes. Without interleaving, fourteen consecutive byte errors would certainly cause a decoding error for a sequence of codewords that can each only correct two-byte errors. However, the figure below shows how the convolutional deinterleaver shown in Figure 22 with  $I = 7$  and  $M = 1$  separates the error burst into isolated pairs of byte errors, where each pair of errors will be corrected by a different codeword.

A sequence of Interleaved Reed-Solomon bytes suffers a noise burst of 14 bytes, shown as red squares.

After deinterleaving, the noisy bytes are distributed over multiple codewords so that they can be corrected.

138. The concepts of interleaving in general and the interleaving of Reed-Solomon data bytes specifically were well known to a POSA long before the alleged invention. For example, in his 1971 journal paper in the IEEE Transactions on Communications, entitled “Burst-Correcting Codes for the Classic Bursty Channel,” G. David Forney discussed using convolutional interleaving with error control coding as a way to overcome bursts of noise. Forney also described the triangular implementation of convolutional interleaving with the constraint that  $D = M \times I + 1$ . In his description he used the variable  $B$ , where  $B = D - 1$ , instead of  $D$  to describe interleaver depth. Forney uses  $B'$  instead of  $M$ , i.e.,  $M = B'$  and  $N$  instead of  $I$  for interleaver blocklength, i.e.,  $I = N$ . Forney identifies the delay of the convolutional interleaver (the delay experienced by all symbols) as  $(N - 1)B$  symbol times, which is  $(I - 1)(D - 1)$  using the notation introduced earlier. Forney’s discussion of the triangular implementation of convolutional interleaving is shown below:

776

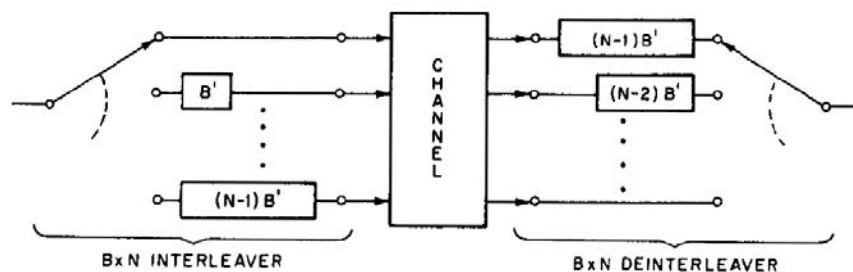


Fig. 2. Periodic interleaver and corresponding deinterleaver ( $B' = B/N$ ).

**Figure 23: Forney's illustration of a convolutional interleaver.**

Schematically, as illustrated in Fig. 2, symbols to be interleaved are arranged in blocks of N (by a serial/parallel conversion, if necessary). The  $i$ th symbol in each block is delayed by  $(i - 1)NB'$

time units through a  $(i - 1) B'$  stage shift register clocked once every  $N$  symbol times, where  $B' = B/N$ . (A time unit thus corresponds to the transmission of a block of  $N$  symbols.) Output bits may be serialized for channel transmission. At the receiver, groups of  $N$  symbols are reblocked, and the  $i$ th symbol in each block is delayed by  $(N - i)NB'$  time units through an  $(N - 1) B'$  stage shift register.

We call this a  $B \times N$  interleaver. Correspondingly there exists a similar but inverse  $B \times N$  deinterleaver, also illustrated in Fig. 2. The combination has the following properties.

- 1) All symbols receive a total delay of  $(N - 1)B'$  time units, or  $N(N - 1)B' = (N - 1)B$  symbol times (plus the channel delay).
- 2) The memory requirements at transmitter and receiver are  $N(N - 1)B'/2$ , or  $N(N - 1)B/2 = (N - 1)B/2$  total.

A single channel burst affecting  $B'$  or fewer blocks ( $B - N + 1$  or fewer symbols) passes through the deinterleaver in such a way as to affect only one of the  $N$  deinterleaver output streams at a time. See Fig. 3. Repeated bursts separated by guard spaces of  $(N - 1)B'$  or more blocks ( $(N - 1)B + N - 1$  symbols) also affect only one of the  $N$  output streams at a time.

- 4) Channel burst affecting  $kB'$  or fewer blocks affects no more than  $k$  of the  $N$  deinterleaver output streams at a time. Repeated bursts separated by guard spaces of  $(N - k)B'$  or more blocks affect only  $k$  of the output streams at a time. See Fig. 4.

(Forney "Burst-Correcting Codes for the Classic Bursty Channel," excerpts from pages 775-776)

139. As described by Forney in items 1) and 2) above, delay and memory requirements are distinct aspects of an interleaver implementation. Delay, i.e. total delay or overall delay or end-to-end delay, is the delay in time that is experienced by all symbols from the time they enter the interleaver to the time they exit the deinterleaver. For convolutional interleavers, this delay is typically  $(I - 1)(D - 1)$ . Memory requirements describe the amount of memory required for a specific implementation of the interleaver, deinterleaver, or interleaver/deinterleaver pair. There are various implementations that use different amounts of memory while still maintaining the same delay. For example, the implementation of Fig. 9-13 of *Fundamentals of DSL Technology* requires

$I \times D$  memory locations for the interleaver. In contrast, the triangular implementation of VDSL1 and Forney ideally uses  $(I - 1)(D - 1)/2$  memory locations (ignoring memory for pointers and non-ideal implementation) for the interleaver, but these implementations have the same delay of  $(I - 1)(D - 1)$  symbols.

140. The memory required for an interleaver with a particular set of parameters  $D$  and  $I$  is implementation specific. More memory than the smallest possible memory is often required by a typical implementation. Thus, to understand which interleaver parameters  $D$  and  $I$  can be supported by a particular transceiver with a specified amount of available memory, more information is needed about the specific implementation used by that transceiver. In contrast, end-to-end delay typically remains the same across implementations. A first transceiver can communicate what end-to-end delay it can support to a second transceiver without the second transceiver needing information about the implementation details to properly understand the interleaver parameters  $D$  and  $I$  that can be supported by the first transceiver. In other words, two transceivers can communicate properly when using convolutional interleavers/deinterleavers with the same  $I$  and  $D$  values, leading to the same delay. The transceivers can, however, use different amounts of memory to implement the respective interleaver or deinterleaver using those  $I$  and  $D$  values (with the resultant delay).

141. U.S. Patent No. 7,269,208 issued to Mazzoni teaches the use of convolutional interleaving with the constraint that  $D = M \times I + 1$ , for example at 2:39-48, shown below:

The interleaving means are then adapted to effect convolutional interleaving of  $I$  branches with  $i-1$  blocks of  $N$  ~~M~~ bytes. The deinterleaving means are adapted to implement convolutional deinterleaving with  $I'$  branches of  $i'-1$  blocks of  $M'$  bytes.  $I$  and  $I'$  are sub-multiples of  $N$ , and  $i$  and  $i'$  are the current relative indexes of the branches. The size in bytes of the first memory space is equal to  $I \times (I-1) \times M/2$ , and the size in bytes of the second memory space is



equal to  $I' \times (I'-1) \times M' / 2$ . The sizes of the two memory spaces are set by  $I$ ,  $I'$ ,  $M$  and  $M'$ . (Mazzoni 2:39-48)<sup>3</sup>

142. Throughout this report we will adopt the convention of Mazzoni that for systems where convolutional interleaving is constrained with depth  $D$  and blocklength  $I$  such that  $D = MI + 1$  so that we will use the parameters  $M$  and  $I$  for interleavers and deinterleavers operating on communication in one direction (for example downstream) and the parameters  $M'$  and  $I'$  for interleavers and deinterleavers operating on communication in the other direction (for example upstream).

143. Mazzoni teaches the triangular implementation of convolutional interleaving for example at 5:9-20

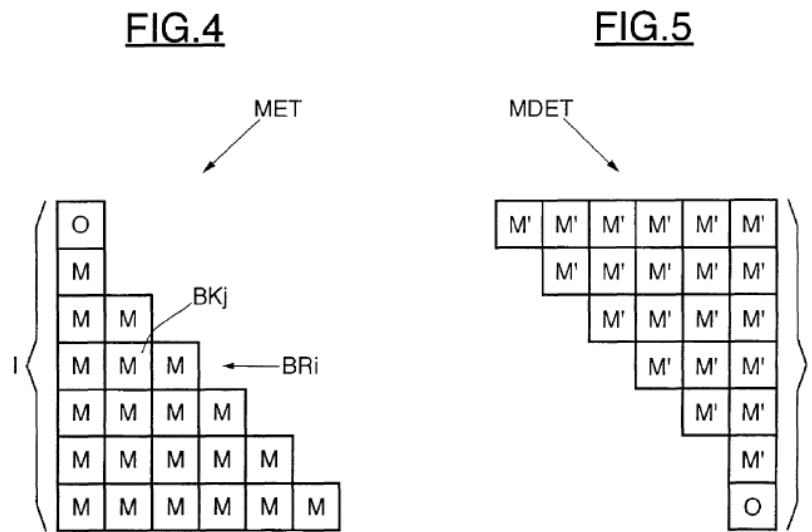
The internal architecture and the operation of the interleaving and deinterleaving means will now be described in more detail with more particular reference to FIGS. 3-8. As shown in FIG. 3, and as already explained, the interleaving means MET follow the Reed-Solomon coding means CRS, and the deinterleaving means MDET precede the ReedSolomon decoding means DCRS. As shown diagrammatically in FIGS. 4 and 5, the interleaving and deinterleaving are convolutional triangular interleaving and deinterleaving. There are  $I$  branches of  $i-1$  blocks of  $M$  bytes for interleaving and  $I'$  branches of  $i'-1$  blocks of  $M'$  bytes for deinterleaving. (Mazzoni 5:9-20)

144. Figures 4 and 5 on drawing sheet 4 of Mazzoni illustrate the triangular implementation of convolutional interleaving:

---

<sup>3</sup> The Mazzoni certificate of correction clarifies that N at 2:41 should be replaced by M.





U.S. Patent Sep. 11, 2007 Sheet 4 of 7 US 7,269,208 B2

**Figure 24: Figures 4 and 5 from Drawing Sheet 2 of Mazzoni illustrate a convolutional interleaver.**

145. Because Mazzoni teaches the specific triangular implementation, the interleaver/deinterleaver depth is always a multiple of the interleaver block length ( $I$  or  $I'$ ) decremented by 1. Specifically, the interleaver depth is  $M \cdot (I - 1)$  and the deinterleaver depth is  $M' \cdot (I' - 1)$ . The interleaver depth is described, for example, at 2:39-43, shown below

The interleaving means are then adapted to effect convolutional interleaving of  $I$  branches with  $i-1$  blocks of  $N$  bytes. The deinterleaving means are adapted to implement convolutional deinterleaving with  $I'$  branches of  $i'-1$  blocks of  $M'$  bytes. (Mazzoni 2:39-43)<sup>4</sup>

146. Under the restriction that the interleaver depth is a multiple of  $(I - 1)$ , and equivalently that the deinterleaver depth is a multiple of  $(I' - 1)$ , and using this specific triangular implementation taught by Mazzoni, the memory is specified in the teachings of Mazzoni as  $\times (I - 1) \times M/2$  for the interleaver and  $I' \times (I' - 1) \times M'/2$  for the deinterleaver. This is taught

<sup>4</sup> The Mazzoni certificate of correction clarifies that  $N$  at 2:41 should be replaced by  $M$ .

for example at 2:45-48, 6:37-43, claim 2 at 8:34-38, claim 9 at 9:47-51, and claim 14 at 10:43-47, all of which are shown below:

The size in bytes of the first memory space is equal to  $I \times (I-1) \times M/2$ , and the size in bytes of the second memory space is equal to  $I' \times (I'-1) \times M'/2$ . The sizes of the two memory spaces are set by  $I$ ,  $I'$ ,  $M$  and  $M'$ . (Mazzoni 2:45-48)

The parameters  $I$ ,  $I'$ ,  $M$  and  $M'$  can be determined from the above capacities. More particularly, the size of the first memory space needed to implement triangular convolutional interleaving with  $I$  branches of  $i-1$  blocks of  $M$  bytes is equal to  $I \times (I-1) \times M/2$ . Similarly, the size of the second memory space ESM2 required to support the uplink bit rate is equal to  $I' \times (I'-1) \times M'/2$ . . (Mazzoni 6:37-43)

2. The device according to claim 1 wherein the size of the first memory space is equal to  $I \times (I-1) \times M/2$  bytes, the size of the second memory space is equal to  $I' \times (I'-1) \times M'/2$  bytes, and the sizes of the first and second memory spaces are set by  $I$ ,  $I'$ ,  $M$  and  $M'$ . (Mazzoni claim 2 8:34-38)

9. The device according to claim 8 wherein the size of the first memory space is equal to  $I \times (I-1) \times M/2$  bytes, the size of the second memory space is equal to  $I' \times (I'-1) \times M'/2$  bytes, and the sizes of the first and second memory spaces are set so by  $I$ ,  $I'$ ,  $M$  and  $M'$ . (Mazzoni claim 9 9:47-51)

14. The method according to claim 13 wherein the size of the first memory space is equal to  $I \times (I'-1) \times M/2$  bytes, the size of the second memory space is equal to  ~~$I' \times (I'-1) \times M'/2$~~   $I \times (I-1) \times M/2$  bytes, and the sizes of the first and second memory spaces are set by  $I$ ,  $I'$ ,  $M$  and  $M'$ . (Mazzoni claim 14 10:43-47)<sup>5</sup>

147. The use of these expressions to calculate the memory size for the interleaver and deinterleaver is further emphasized by the example that is described at 6:37-50.

The parameters  $I$ ,  $I'$ ,  $M$  and  $M'$  can be determined from the above capacities. More particularly, the size of the first memory space needed to implement triangular convolutional interleaving with  $I$  branches of  $i-1$  blocks of  $M$  bytes is equal to  $I \times (I-1) \times M/2$ . Similarly, the size of the second memory space ESM2 required to support the

<sup>5</sup>  ~~$I' \times (I'-1) \times M'/2$~~  is replaced by  $I \times (I-1) \times M/2$  according to the Mazzoni certificate of correction.

uplink bit rate is equal to  $I \times (I-1) \times M'/2$ . Also,  $I$  and  $I'$  must be sub-multiples of the size  $N$  of the Reed-Solomon code.

Since  $I \times (I-1) \times M'/2$  must be equal to 24,960, it is possible to choose  $I=40$  and  $M=32$ . Similarly, because  $I' \times (I'-1) \times M'/2$  must be at least equal to 1,920, it is possible to choose  $I'=24$  and  $M'=7$ . This requires a slight increase in size to 1,932 to facilitate the implementation. The final size of the memory  $MM$  is therefore equal to 26,892 bytes. (Mazzoni 6:37-50)

#### **D. Shared Memory**

148. Computing devices have long utilized common memory used by at least two functions. In fact, computing devices have also long utilized common memory used by at least two functions, where a portion of the memory can be used by either one of the functions. Memory allocation using a common memory resource is the common practice of general purpose computers and specialized computers such as digital signal processing devices, which feature a common memory resource used for (and shared by) multiple applications as part of their structure.

149. The idea that a computer will have a common memory resource that is shared by multiple functions is evident in the standard libraries defined for the C programming language as described in the 1988 Second Edition of the classic 1978 book “The C Programming Language” by B. W. Kernighan and D. M. Ritchie. On page 167, as shown below, this classic book on computer programming defines the functions `malloc` and `calloc` that obtain blocks of memory dynamically from the common memory resource.

**7.8.5 Storage Management**

The functions `malloc` and `calloc` obtain blocks of memory dynamically.

```
void *malloc(size_t n)
```

returns a pointer to `n` bytes of uninitialized storage, or `NULL` if the request cannot be satisfied.

```
void *calloc(size_t n, size_t size)
```

returns a pointer to enough space for an array of `n` objects of the specified size, or `NULL` if the request cannot be satisfied. The storage is initialized to zero.

The pointer returned by `malloc` or `calloc` has the proper alignment for the object in question, but it must be cast into the appropriate type, as in

```
int *ip;
```

```
ip = (int *) calloc(n, sizeof(int));
```

`free(p)` frees the space pointed to by `p`, where `p` was originally obtained by a call to `malloc` or `calloc`. There are no restrictions on the order in which space is freed, but it is a ghastly error to free something not obtained by calling `calloc` or `malloc`.

150. The function `free` shown above un-allocates the memory block that was allocated by `malloc` or `calloc`. As defined in this ubiquitous computer programming language, at least a portion of the memory may be allocated to any function at any one particular time. Indeed, which portion of the memory is used to supply the allocated memory is not even visible to the calling function, it only sees the pointer to the memory resource.

151. Many graduate students and working engineers in the 1980s and 1990s, including myself, implemented standard communications tools including multicarrier transmission, Reed-Solomon coding, and interleaving using the C programming language and in particular the `malloc`, `calloc`, and `free` commands to manage memory usage carefully.

152. Variables and arrays can often be defined even without explicitly using these commands, but these variables and arrays are nonetheless allocated memory from the common memory resource which is returned to the common memory resource when the program has completed its execution. Here again, which portion of the memory is used to supply the allocated memory is not even visible to the programmer, who only sees the variable or array names that

indirectly refer to the memory resource. Here again, at least a portion of the memory may be allocated to any function at any one particular time.

153. After initial simulations that were often implemented using the C programming language, standard communications tools including multicarrier transmission, Reed-Solomon coding, interleaving, and retransmission were implemented on digital signal processors (DSPs) such as the AT&T DSP (1981), the AT&T DSP 32 (1986), and the Texas Instruments TMS 320 (1994). These digital signal processing chips all featured common memory used by at least two functions, where a portion of the memory can be used by either one of the functions.

154. The AT&T DSP was described by J. R. Boddie in his 1981 article in the Bell System Technical Journal. As shown below as XXX of this document, Figure 2 on page 1433 of Boddie's 1981 article shows that the DSP has a single common resource for data memory so that the various functions using that data memory had to share this common resource and at least a portion of the memory may be allocated to any function at any one particular time.

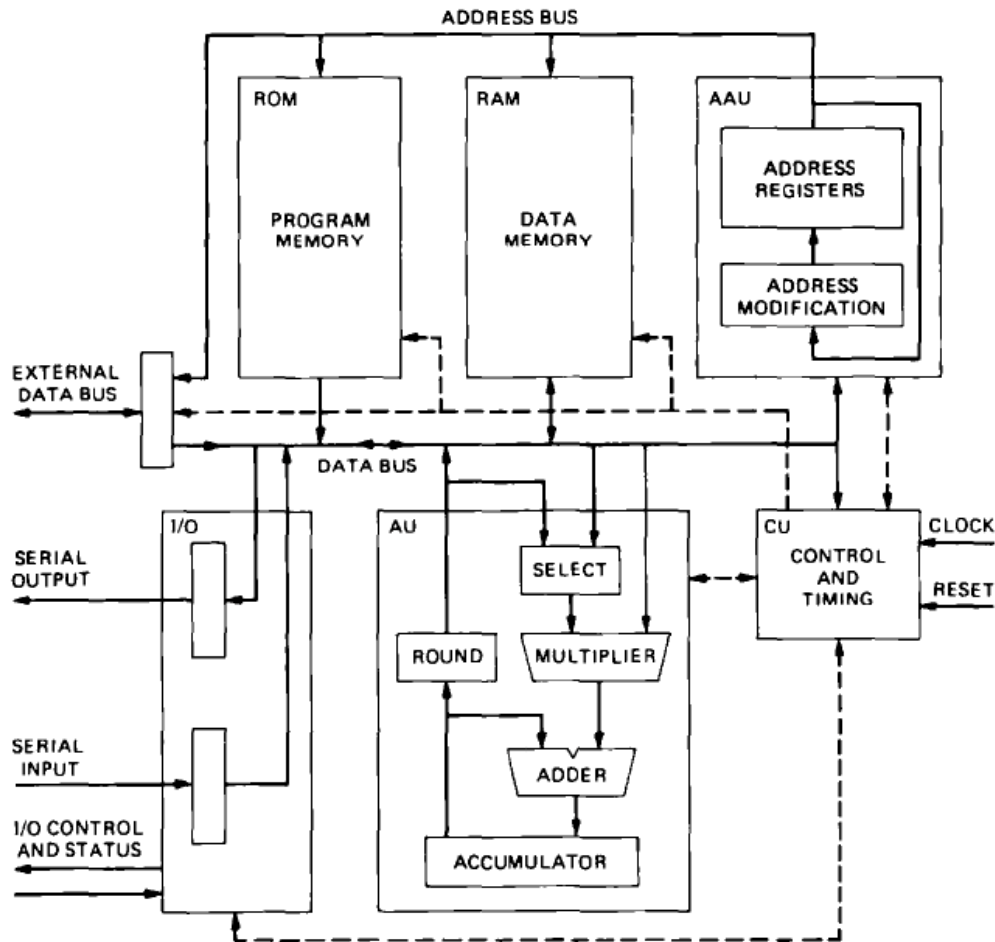


Fig. 2—Digital signal processor block diagram.

OVERVIEW 1433

**Figure 25:** This figure, which is Figure 2 from the 1981 Boddie journal paper, shows the AT&T DSP architecture which features a single common resource for data memory so that the various functions using that data memory had to share this common resource and at least a portion of the memory may be allocated to any function at any one particular time.

155. In his 1981 article and other articles in that issue of the Bell System Technical Journal, Boddie described how the original AT&T DSP could support a variety of applications, shown below in Table 1 of that article. Boddie revisited that historic device in his 2017 magazine article “A Brief History of AT&T’s First Digital Signal Processor.”

**Table I—Application complexity of DSP**

Application	Complexity (No. of DSPs)
Second-order section	0.03
ADPCM* coder	0.25
Dual-tone receiver	0.90
Modem, 1200-baud	1.50
Transmultiplexer	6.00

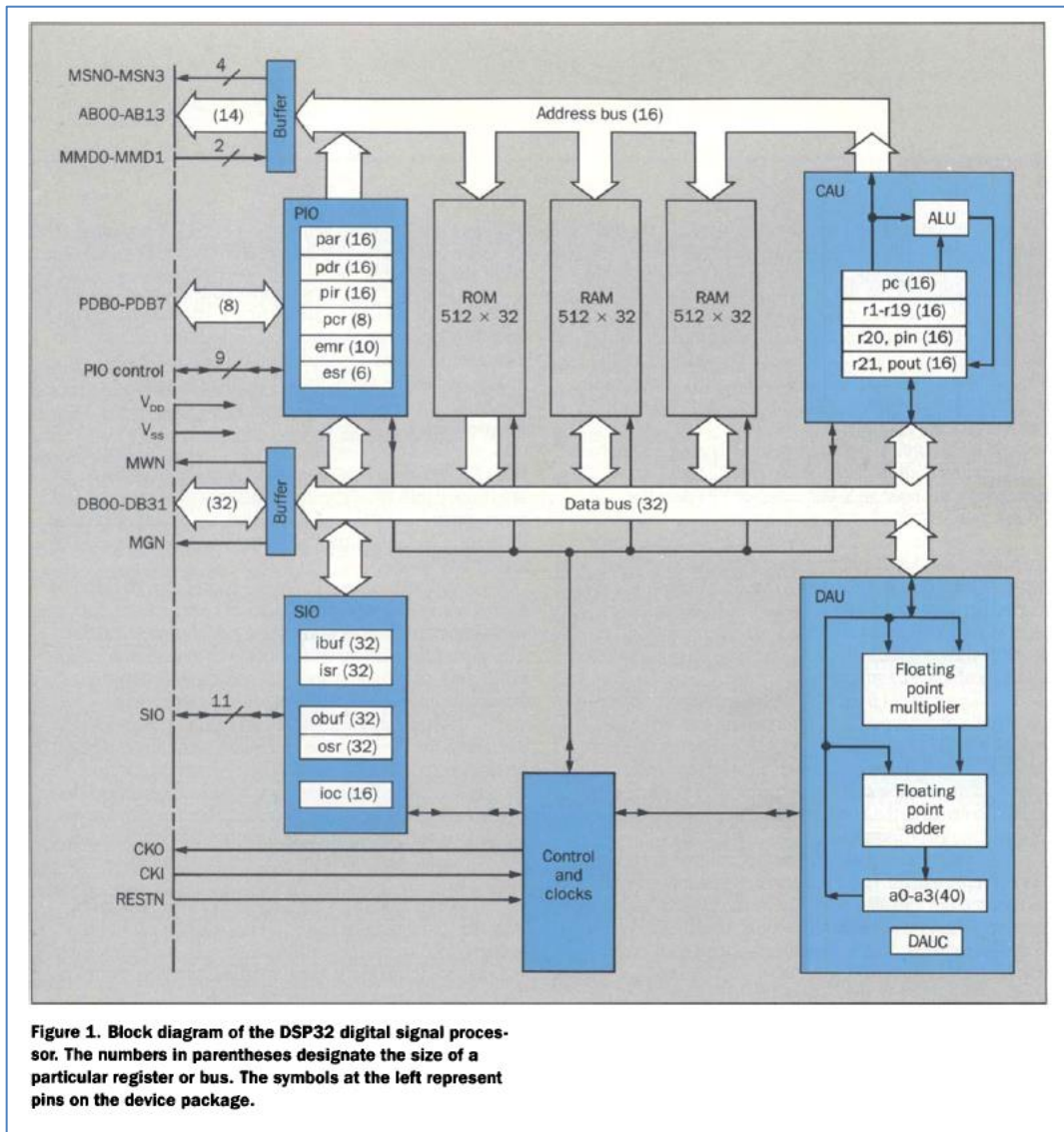
\* Adaptive differential pulse-code modulation.

156. The original DSP used fixed point arithmetic. In 1986, Boddie described the AT&T DSP32, which was a digital signal processor that supported 32-bit floating-point arithmetic. As described by Boddie in his *AT&T Technical Journal* article, entitled “The DSP32 Digital Signal Processor and Its Application Development Tools,”

The WE® DSP32 digital signal processor is a high-speed, programmable, VLSI circuit with 32-bit floating-point arithmetic. The device can be used cost-effectively in a wide variety of complex digital signal processing applications, such as speech recognition, high-speed modems, low bit-rate voice coders, multi-channel signaling systems, and signal processing workstations. In this paper, we review the architecture of the DSP32 and present its instruction set, including some examples. We also discuss the software and hardware support tools that are available for developing DSP32 applications.

157. As with the DSP, the DSP32 used a common resource for data memory so that the various functions using that data memory had to share this common resource and at least a portion of the memory may be allocated to any function at any one particular time. Figure 1 from page 93 of Boddie’s 1986 article shows the DSP 32 architecture. This figure is reproduced as XXX below:





**Figure 26: Reproduction of Figure 1 from the 1986 Boddie journal paper, shows the AT&T DSP32 architecture which features a common resource for data memory so that the various functions using that data memory had to share this common resource and at least a portion of the memory may be allocated to any function at any one particular time.**

158. The Fast Fourier Transform (FFT), a basic building block of DMT or OFDM, could be performed on a DSP32 in 1986 at a throughput of 1024 received symbols every 19.2 ms or 53,333 symbols per second. Thus, the FFT function would use portions of the shared memory in the RAM. This is described, for example, in Boddie's 1986 article:

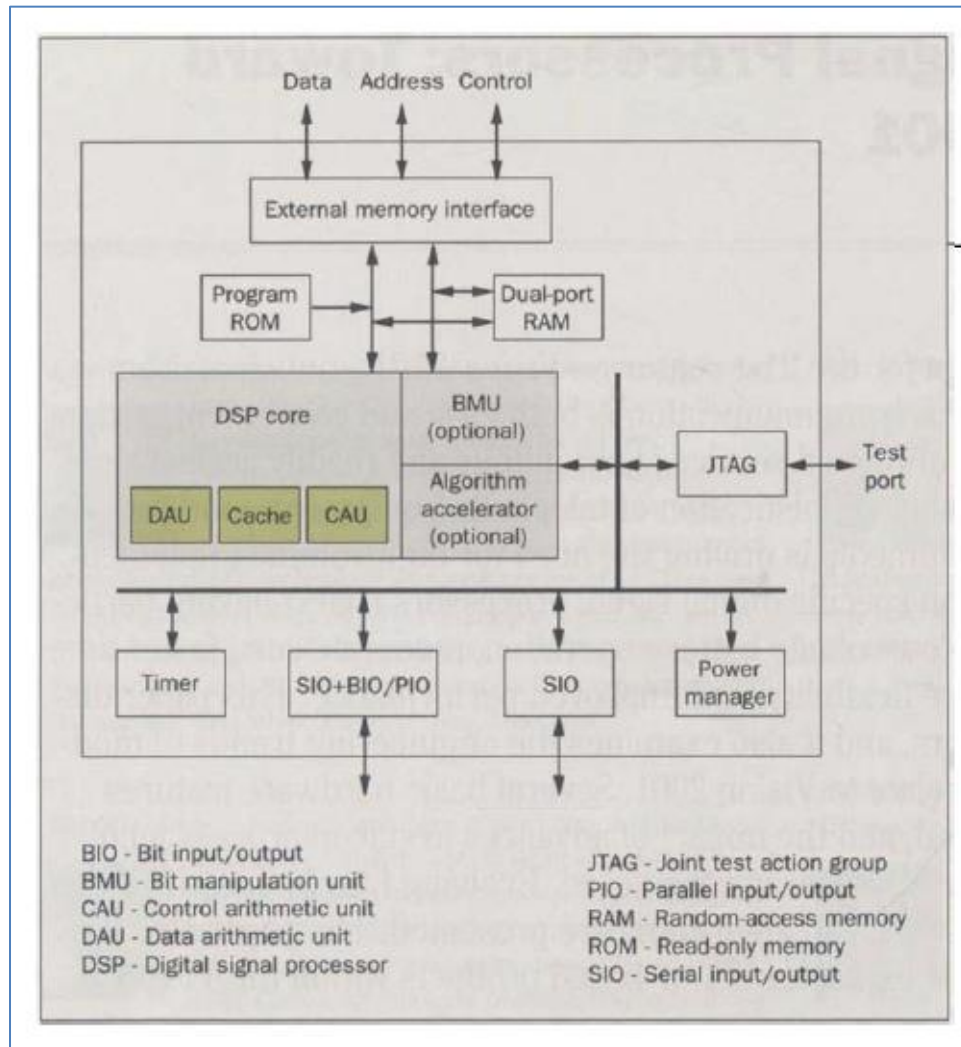


The DSP32 can do the following, all with 32-bit, floating-point precision and dynamic range:

- a 1024-point, complex, Fast Fourier Transform (FFT) in 19.2 ms (including bit reversal)
- a finite impulse response filter in 250 ns per tap
- a second-order section (four multiply) for a recursive, infinite-impulse-response (IIR) filter in 1  $\mu$ s.

159. In 1995 R. E. Crochiere and J. R. Boddie published a journal paper in the *AT&T Technical Journal* entitled “Digital Signal Processors: Toward Vision 2001” that described the general trend towards the use of digital signal processors with a shared memory resource. In their words, “Today, the increasing sophistication of telephone equipment, modems, and computer multimedia is driving the need for high-volume implementation of application-specific digital signal processors (DSPs) having particular emphasis on low-voltage battery operation, moderate cost, faster time to market, software flexibility, and improved performance.”

160. Figure 27, shown below, reproduces Figure 1 from the Crochiere and Boddie 1995 article, which shows the general architecture of DSPs. Note that the module in the diagram labeled Dual-port RAM is the shared memory resource. Any DSP that follows this general architecture will have the property that at least a portion of the memory may be allocated to any function at any one particular time.



**Figure 1.** This illustration shows some of the basic features found in DSP architectures. DSPs are designed for high-speed, real-time numerical computation. They differ from conventional microprocessors or microcontrollers in that they are designed specifically for signal processing applications. They can process signals far more cost effectively—and with lower overhead and power consumption—than general-purpose processors. They also feature input/output (I/O) interfaces configured for such sampled data-processing systems as coders/decoders (codecs) and high-speed memory.

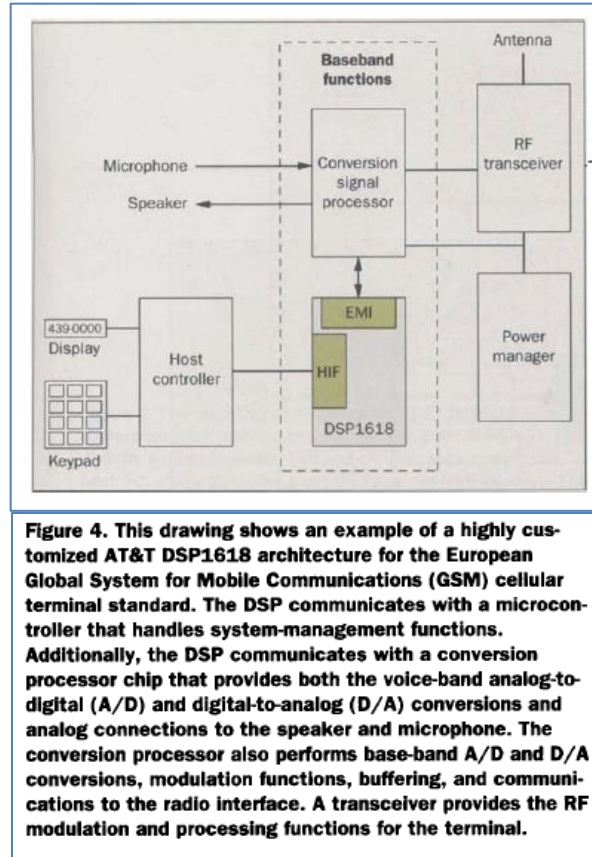
Figure 27: Reproduction of Figure 1 from the 1995 Crochiere and Boddie journal paper, shows the general architecture for DSPs. The module in the diagram labeled Dual-port RAM is the shared memory resource.

161. Not only is the memory of the DSP shared, but it is also recognized by Crochiere and Boddie in 1995 as that DSPs would have only the minimum amount of this precious resource, so that it must be allocated carefully. Note, however, that a DSP can implement multiple functions that run simultaneously and can also be reprogrammed to implement a different function than the functions previously implemented. For a DSP to function as intended, it is necessary that the memory can support the operation of multiple functions simultaneously and that a portion of the RAM can be used by more than one function.

**Memory.** Two types of memory are typically incorporated into the design of DSPs: *read-only memory* (ROM) for non-volatile storage of program instructions, and *random-access memory* (RAM) for storage of algorithm-state variables, computed data, or control information. RAM is also used for applications in which programs are downloaded or cache loaded into the DSP.

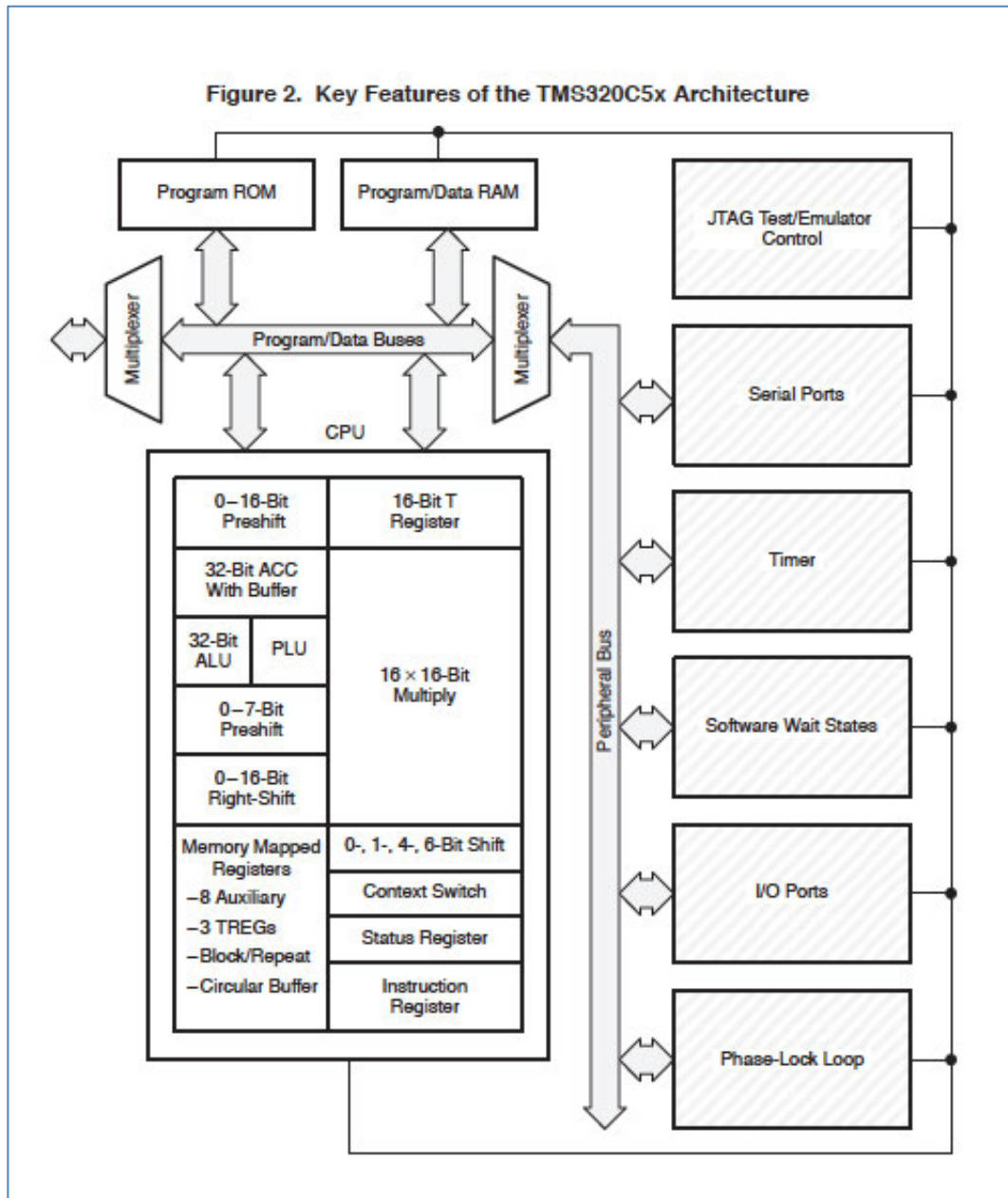
For high-volume, cost-sensitive applications, DSPs are often customized. They are designed having only the minimum required RAM and ROM, as well as the necessary I/O interfaces for a given application. Such a design reduces both the DSP's silicon area and device cost. Memory sizes range from 512 to 8,192 words of RAM and from 4,096 to 32,767 words of ROM.

162. The Crochiere and Boddie 1995 article gives the example of a DSP 1618 performing the digital baseband functions for both the 13-kbps GSM speech coding and the channel processing. The channel processing functions include such operations as Viterbi processing, convolutional decoding, and burst interleaving. Figure 28 shows the overall GSM architecture using the DSP 1618. As with other DSPs, in the DSP 1618 at least a portion of the memory (the 512 to 8192 words of RAM used for storage of algorithm-state variables, computed data, or control information) may be allocated to any of several functions at any one particular time.



**Figure 28:** A reproduction of Figure 4 from the 1995 Crochiere and Boddie journal paper, shows the architecture for a DSP1618. The DSP 1618 performs the digital baseband functions for both the 13-kbps GSM speech coding and the channel processing.

163. While AT&T was developing its line of DSPs, there was serious competition from Texas Instruments, who developed their own line of DSPs including the extremely successful TI DSP TMS320. In 1994, TI published an application book entitled “Telecommunications Applications With the TMS320C5x DSPs,” which described transceivers with shared memory. Figure 29 below shows Figure 2 from the application book, which illustrates the TMS 320 C5x Architecture, which features the Program/Data RAM as the shared memory.



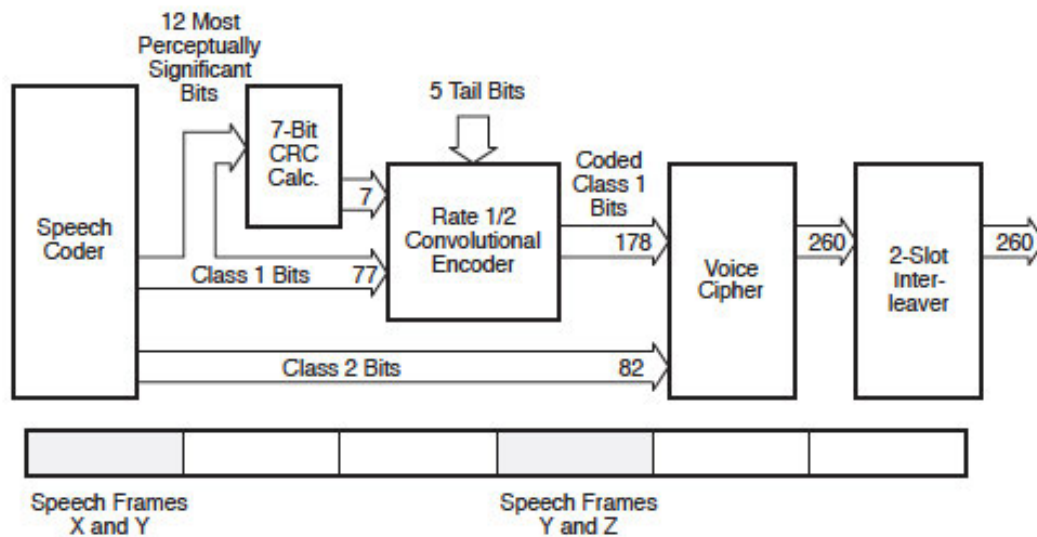
**Figure 29: Reproduction of Figure 2 from Telecommunications Applications With the TMS320C5x DSPs.**

164. The application book describes an IS 54 transceiver, and also describes a C-language simulation of IS-54 that TI made publicly available. As discussed above, any C-language implementation of a transceiver would utilize shared memory. See the excerpt and figure shown below.

### Introduction

This paper describes a C language simulation of both the transmit and receive baseband processing for a digital cellular telephone that meets the U.S. digital cellular standard (IS-54B). This simulation is needed for two reasons: first, to gain greater understanding of the IS-54 digital cellular standard and the associated digital signal processing required in a terminal that meets this standard with a vision toward efficient implementation on the TMS320 DSPs; second, to gain the capability to evaluate the effect of bit errors on the speech coder (vector sum excited linear prediction, or VSELP) and IS-54 control functions. This necessitated development of a simulation of the IS-54 processing and RF channel. See Figure 1.

**Figure 2. IS-54 Error Encoding and Interleaving**

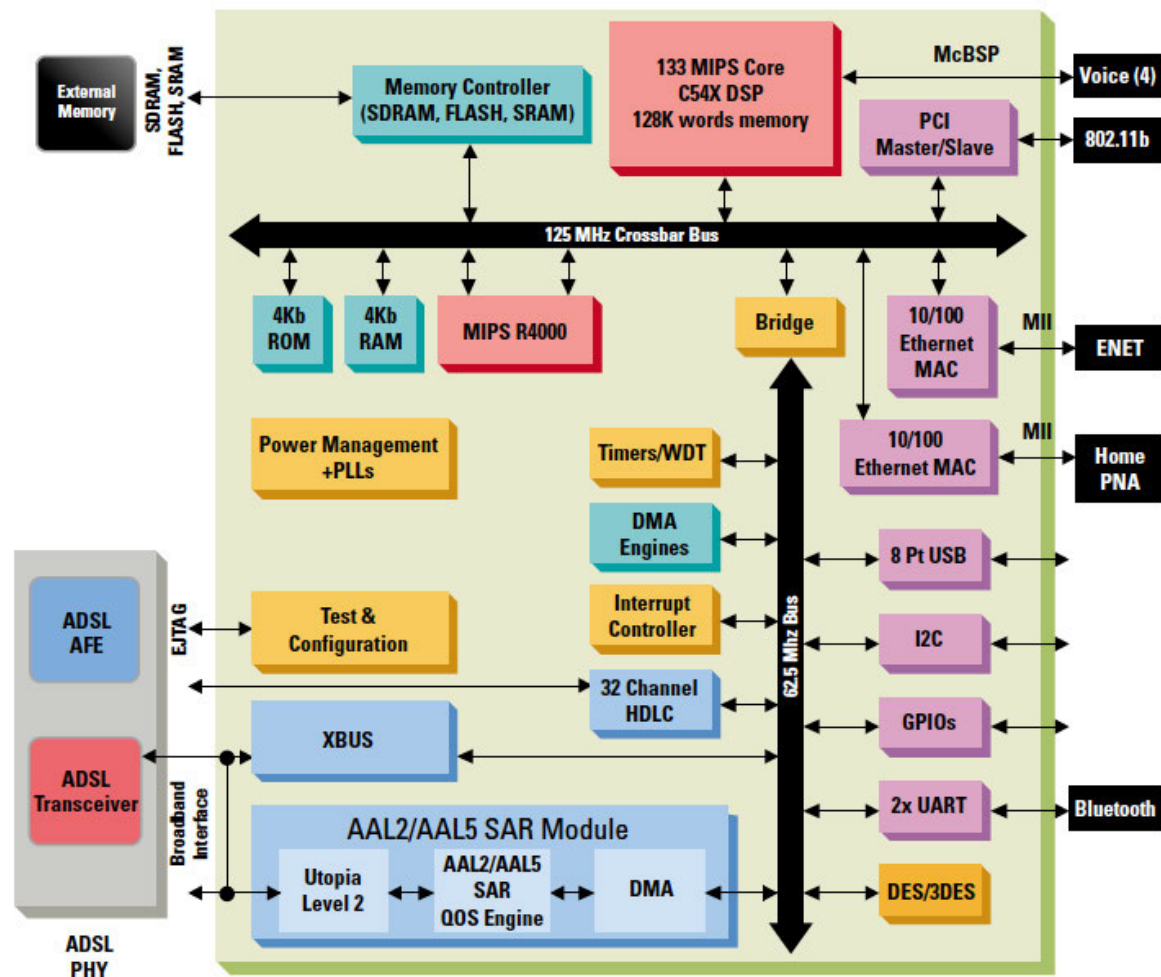


165. Additionally, the TMS 320 was specifically used to implement ADSL transceivers with shared memory long before the alleged inventions. For example, the TI AR5 Chipset Family described in the 2001 Product Bulletin entitled “AR5 Chipset Family: ADSL Router, Integrated Access Device and Residential Gateway Chipset Solutions” included an ADSL transceiver shown in the figure below:



# AR5 Chipset Family

## ADSL Router, Integrated Access Device and Residential Gateway Chipset Solutions



### AR5V10 DSL Communications Processor

*A superior architecture, including a 32-bit RISC processor and a high-performance DSP, provides outstanding throughput for best-in-class routing and low voice latency.*

166. As described in the product bulletin as excerpted below, the ADSL transceiver (specifically the TNET5100) is implemented on a TMS 320 DSP, specifically the TMS320C6000,

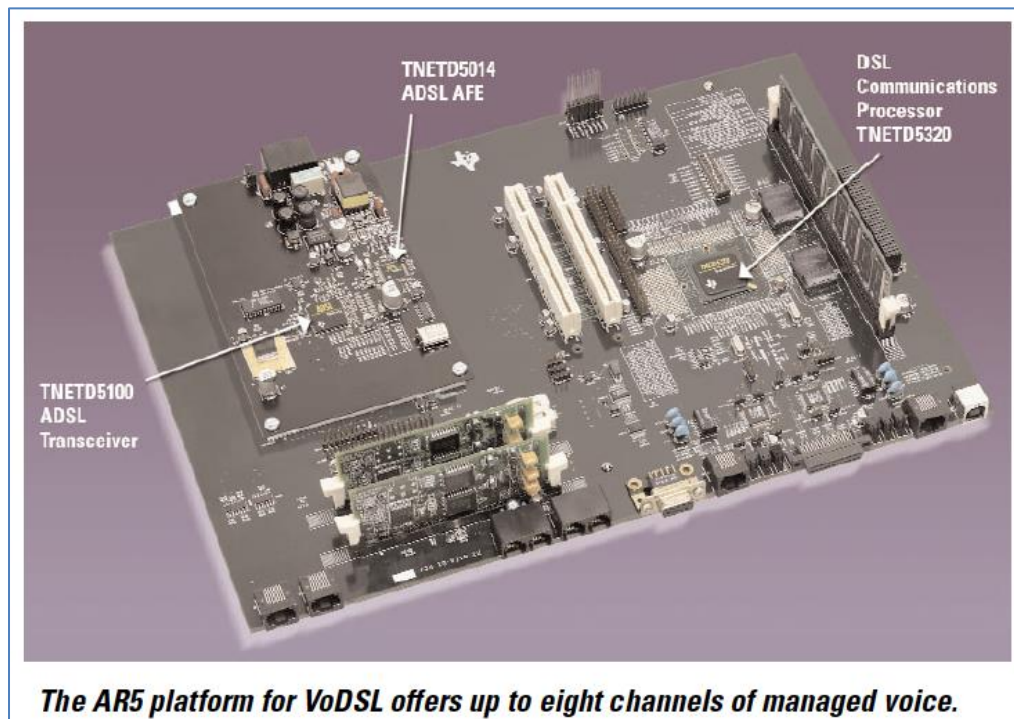


which is explicitly identified as a programmable DSP-based engine, which would use shared memory for the programmable functions.

The common two-chip AX5 PHY includes:

- The TNETD5100 ADSL Transceiver, a programmable TMS320C6000™ DSP-based engine dedicated to high-speed ADSL transmission and reception.
- The TNETD5014 ADSL Codec and Line Driver/Receiver, a highly integrated ADSL analog front end (AFE) with support for both full-rate G.dmt and G.lite.

167. Below is a photograph from the AR5 product bulleting showing the TNET 5100 ADSL Transceiver.



168. The AR5 product bulletin explicitly describes the shared memory used by the TNETD5100 ADSL Transceiver as a 1-Mbit on-chip instruction/data cache with enhanced DMA control for sustained high throughput, as shown below:

<b><i>PHY Devices (common to all chipsets)</i></b>	
<b><i>TNETD5100 ADSL Transceiver</i></b> <ul style="list-style-type: none"> <li>• Programmable 1600-MIPS TMS320C6200™ DSP core with VelociTI™ very-long instruction word (VLIW) architecture performs up to eight 32-bit instructions simultaneously</li> <li>• 1-Mbit on-chip instruction/data cache with enhanced DMA control for sustained high throughput</li> <li>• Glueless 32-bit external memory interface to synchronous and asynchronous devices</li> <li>• Glueless interface to TNETD5014 AFE</li> </ul>	<b><i>TNETD5014 ADSL Codec and Line Driver/Receiver</i></b> <ul style="list-style-type: none"> <li>• Complete AFE integrates 14-bit ADC and DAC, line driver/receiver, TX/RX filter, programmable gain amplifiers and equalizer</li> <li>• Supports both full-rate G.dmt (ITU G.992.1) Annex A, and G.lite (ITU G.992.2) ADSL transmission</li> <li>• Coexists with Bluetooth™, Wireless Ethernet (802.11b) and HPNA devices</li> <li>• Low power consumption</li> </ul>

169. Typical implementations of cellular technology use a baseband processor that employs shared memory. Examples include the Zyray spinner baseband processor that supported 3GPP WCDMA technology such as that disclosed in TR 25.896. A press release from April 2004 describes the WCDMA baseband processor as supporting 3GPP Release 99 and providing an upgrade path to release 4.

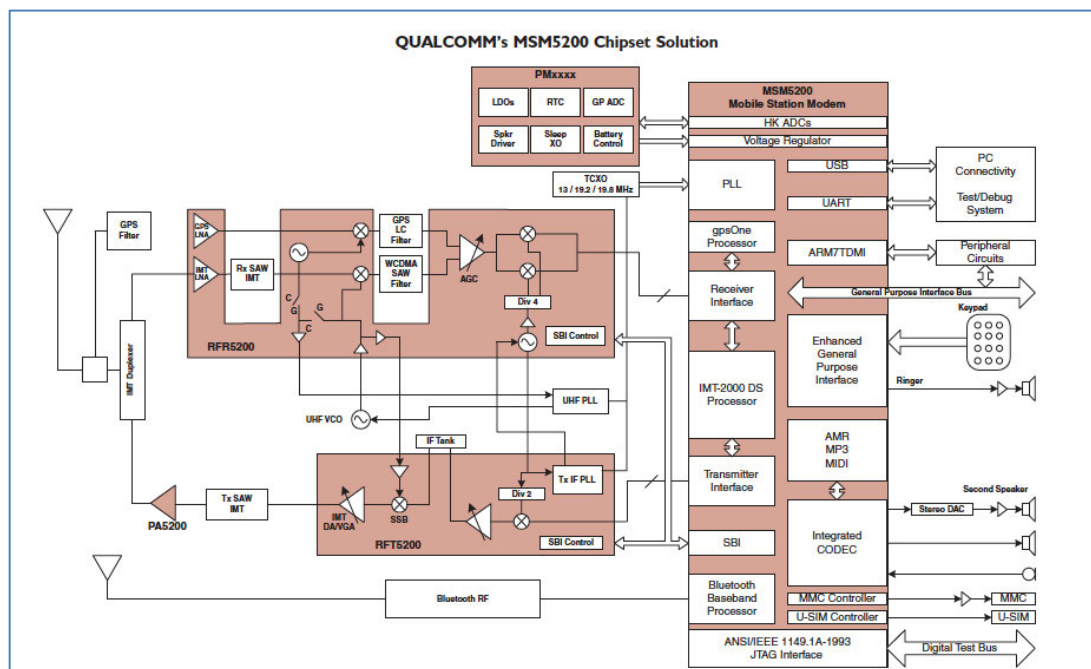
## WCDMA Baseband Processor

Wed, 04/07/2004 - 11:04am by

Zyryl introduces the SPINNERchip 1.1. The product enables device manufacturers to develop single-mode WCDMA, dual-mode GSM/GPRS & WCDMA, and multi-mode GSM/GPRS, EDGE & WCDMA devices. SPINNERchip 1.1 is housed in a 10 x 10 mm BGA including integrated analog technology and is flexible enough to mate with various RF transceivers, baseband processors, and dual-mode protocol stacks. Other features include the following: 3GPP Rel 99 December 2003 compliant WCDMA FDD modem; firmware upgrade path via to 3GPP Rel 4; 384 kbps class modem operation in both uplink and downlink; supports Circuit-Switched Voice, Packet-Switched Data and Circuit-Switched Data; Generic Layer 1 Abstraction Layer allows for 3rd party protocol stack support; complete firmware suite including PHY layer and Layer 1; support for USIM interface (optional); on-chip, programmable power management; supply voltage: 1.8 V / 3.0 V.


Zyray Wireless

170. Another example is the Qualcomm Mobile Station Modem (MSM) 5200 baseband processor chip that was available by 2001 and was 3GPP Release 99 compliant. Shown below is a figure and corresponding text excerpt from Qualcomm’s product literature about the MSM 5200:





**MSM5200 DEVICE DESCRIPTION**



The MSM5200 chipset and system software builds on the successful architecture of QCT's MSM3xxx and MSM5xxx families of CDMA devices, and is based on QCT's core CDMA systems expertise gained from years of field experience with this technology. The MSM5200 solution integrates both digital and analog functions into a single chip.

The MSM5200 solution includes advanced technologies such as GPS-based position location and Bluetooth, as well as multimedia features such as Qtunes™ MP3 player software and Compact Media Extension™ (CMX™) MIDI-based multimedia software. Along with an optimized software solution for the WCDMA modem, QCT offers system development software, verification, test, debug, calibration, manufacturing, and field test support using the WCDMA designer development tools, which reduce time-to-market for a complete WCDMA product.

**MSM5200 General Device Enhancements**

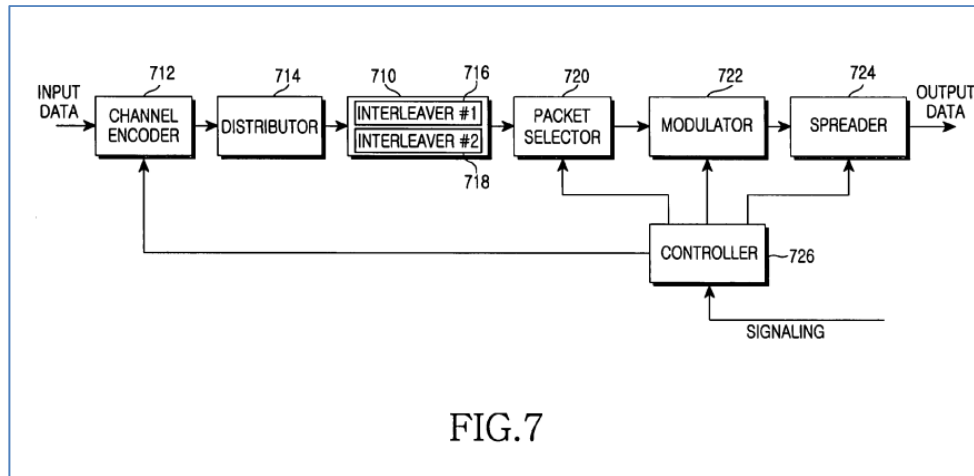
<ul style="list-style-type: none"> <li>• Integrated audio codec</li> <li>• Supports low-power, low-frequency crystal to enable TCXO shutoff</li> <li>• Enhanced I/O support for faster RS-232</li> </ul>	<ul style="list-style-type: none"> <li>• 16-bit-wide memory support</li> <li>• Embedded Bluetooth baseband processor</li> <li>• Voice Recognition (optional)               <ul style="list-style-type: none"> <li>–Speaker-dependent, speaker-independent and</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>voice-prompt support               <ul style="list-style-type: none"> <li>–Multiple-language support</li> </ul> </li> <li>• Integrated mass-storage device (MMC) controller</li> <li>• Audio enhancement technology support (Qtunes and</li> </ul>	<ul style="list-style-type: none"> <li>CMX software)</li> <li>• Integrated USB device controller for fast and simple PC interconnect</li> <li>• Interfaces with the MSP™ family of applications processors</li> </ul>
--	--	---	---

**MSM5200 IMT-2000 DS Device Enhancements**

<ul style="list-style-type: none"> <li>• 3GPP Release 99 compliant</li> <li>• Full 3GPP protocol stack including L1, L2:MAC/RLC/BMC/PDCP,</li> </ul>	<ul style="list-style-type: none"> <li>L3:RRC, (G)MM, SS, CC, SMS, SM, RABM</li> <li>• Data rates: 384 kbps on both the uplink and downlink</li> </ul>	<ul style="list-style-type: none"> <li>• Adaptive Multiple Rate (AMR) speech codec</li> <li>• Support for convolutional and turbo coding</li> </ul>	<ul style="list-style-type: none"> <li>• USIM interface</li> </ul>
--	--	---	--

171. As discussed above, evidence abounds that a POSA understood that for each transceiver function, appropriate memory must be allocated to that function. The very nature of general purpose computers and digital signal processors demands this. Of interest is evidence that a POSA would have understood that this is specifically true for interleaving/de-interleaving, for example in a communications system.

172. As a further example of sharing memory, consider U.S. Patent No. 7,027,782 to Moon et. al. (“Moon”), which illustrates the use of shared memory in a communications system, particularly a mobile communications system. Moon issued on April 11, 2006, from an application filed on October 18, 2002. See Fig. 7 from Moon shown below.



173. Moon notes that while “the first interleaver 716 and the second interleaver 718 are separated by hardware” in Figure 7, “the first interleaver 716 and the second interleaver 718 can be simply logically separated.” Moon at TQD-ADT 070866, 12:39-43. As described by Moon, the “logical separation means dividing one memory into a memory area for storing the systematic bits and another memory area for storing the parity bits” that are interleaved by the first and second interleavers, respectively. Moon at TQD-ADT 070866, 12:43-45. In other words, Moon describes a common memory that is shared (logically separated) between two different functions, e.g.: the first and second interleavers.

174. Moon further describes how the relative sizes, i.e. allocations, of the logically separated memory, i.e. shared memory, may change during the operation of the transceivers as they adjust their operating modes. For example, depending on the coding rate applied, the relative inputs to the first and second interleaver memories can change. With a coding rate of  $\frac{1}{2}$ , the first and second interleaver memories will receive a symmetric input, i.e., for each bit that goes into the first interleaver, one bit will go into the second interleaver. Moon at TQD-ADT 070866, 12:12-14. But for a coding rate of  $\frac{3}{4}$ , for every three bits that go into the first interleaver, only one bit will go into the second interleaver. Moon at TQD-ADT 070866, 12:15-18.

175. Thus, the required relative sizes of the first interleaver and the second interleaver may change as the coding rate changes. With a coding rate of  $\frac{1}{2}$ , 50% of the common memory used for interleaving would be assigned to the first interleaver and 50% of the common memory used for interleaving would be assigned to the second interleaver. With a coding rate of  $\frac{3}{4}$ , 75% of the common memory used for interleaving would be assigned to the first interleaver and 25% of the common memory used for interleaving would be assigned to the second interleaver.

176. Thus, Moon represents a particular application of the well-known and widely employed principle of memory sharing described above.

177. U.S. Patent No. 7,269,208 to Mazzoni et al. (Mazzoni) also illustrates the use of shared memory in communications system. Mazzoni issued September 11, 2007, from a PCT application filed on July 11, 2001. The PCT application published January 24, 2002, and the corresponding U.S. application published on January 30, 2003.

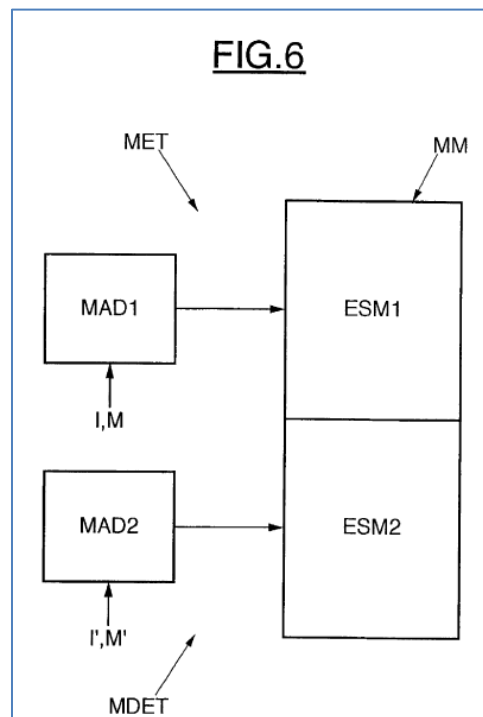
178. Mazzoni is directed to a “device for sending/receiving digital data” that “may include a channel coding/decoding stage including an interleaver, a deinterleaver, and a memory whose minimum size is fixed as a function of maximum bit rate of [a] group of predetermined bit rates.” Mazzoni at Abstract. Mazzoni’s invention “may be advantageously applied to a very high rate digital subscriber line (VDSL) environment. Mazzoni at 1:19-21.

179. Mazzoni notes the well-known fact that the “process of interleaving and deinterleaving data sent and received by a modem necessitates the use of memories.” Mazzoni at 1:39-40. Mazzoni sought to design a modem “which requires a reduced quantity of memory.” Mazzoni at 1:47-48.

180. To do so, Mazzoni then explains how the memory can be shared between the interleaver and deinterleaver. Mazzoni describes the use of a “memory means” that “can be shared

between the interleaving means and the deinterleaving means, and whose memory allocation can be reconfigured in accordance with the bit rate actually processed by the . . . [modem[].” Mazzoni at 1:59-65. “The memory . . . has a first memory space assigned to the interleaving means and a second memory space assigned to the deinterleaving means. The size of each of the two memory spaces is set as a function of the bit rate actually processed by the device.” Mazzoni at 2:10-15.

181. The common memory is depicted in Figure 6 of Mazzoni. Figure 6 shows that “the interleaving means and the deinterleaving means include common memory means MM, e.g., a dual-port random access memory. The memory space of the memory MM is then divided into a first memory space ESM1 assigned to the interleaving means MET, and a second memory space ESM2 is assigned to the deinterleaving means MDET.” Mazzoni at 5:61-67.

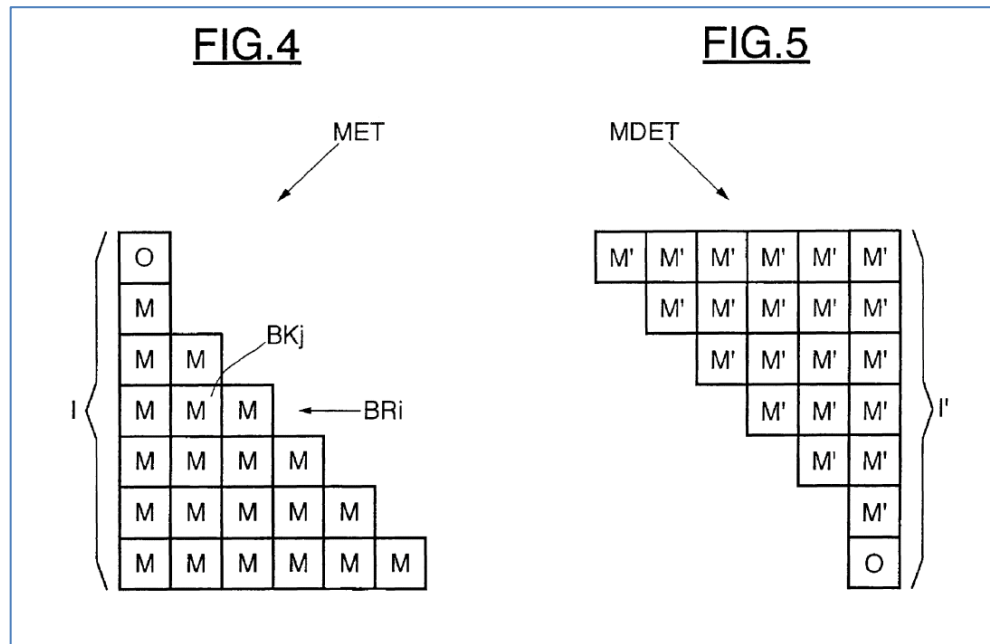


182. Mazzoni also describes the parameters, I and M (associated with the downstream interleaver) and I' and M' (associated with the upstream deinterleaver, respectively) that “define the sizes of the respective memory spaces assigned to the interleaving means and to the



deinterleaving means. This is done according to the bit rate of the information sent by the terminal TO (parameters  $I$  and  $M$ ) and the bit rate of the information received by the terminal TO (parameters  $I'$  and  $M'$ ).” Mazzoni at 5:21-30.

183. As shown below in Figures 4 and 5 from Mazzoni, the typical triangular implementation of convolutional interleaving is used to compute how much memory will be allocated the interleaver and de-interleaver.



184. This is the implementation considered in VDSL1 as discussed in Section X.C below. Mazzoni at 6:37-50. Mazzoni discloses supporting a variety of parameters  $M$  and  $I$ , and uses the same equation for computing the memory requirement of this triangular implementation as disclosed in VDSL1, specifically  $M \times I \times (I - 1)/2$  bytes of memory are required to implement an interleaver or deinterleaver with parameters  $M$  and  $I$ . *id.*

The parameters  $I$ ,  $I'$ ,  $M$  and  $M'$  can be determined from the above capacities. More particularly, the size of the first memory space needed to implement triangular convolutional interleaving with  $I$  branches of  $i-1$  blocks of  $M$  bytes is equal to  $I \times (I-1) \times M/2$ . Similarly, the size of the second memory space ESM2 required to support the uplink bit rate is equal to  $I' \times (I'-1) \times M'/2$ . Also,  $I$  and  $I'$  must be sub-multiples of the size  $N$  of the Reed-Solomon code.

Since  $I \times (I-1) \times M/2$  must be equal to 24,960, it is possible to choose  $I=40$  and  $M=32$ . Similarly, because  $I' \times (I'-1) \times M'/2$  must be at least equal to 1,920, it is possible to choose  $I'=24$  and  $M'=7$ . This requires a slight increase in size to 1,932 to facilitate the implementation. The final size of the memory MM is therefore equal to 26 892 bytes.

185. Thus, Mazzoni describes a common memory that can be allocated between two different functions: the interleaver and the deinterleaver. And the sizes of the interleaver and deinterleaver within that common memory can be reconfigured such that the relative sizes of the interleaver and deinterleaver vary.

186. United States Patent Application US 2004/0165658 A1 (NOK00732238) filed February 4, 2004 and published August 26, 2004 by Albert Van Hoogenbemt provides another example of memory sharing specifically in the context of DSL communication. As described in the Abstract below, Hoogenbemt teaches a DSL telecommunication device comprising a common memory that is shared between the FFT and the Demapper if the downstream path and the mapper and IFFT of the upstream path.

(57)

**ABSTRACT**

A Digital Subscriber Line [DSL] telecommunication device comprising at least one common memory (CM) that is shared between circuits (FFT, Demapper) of the downstream path and corresponding circuits (Mapper, IFFT) of the upstream path. The shared or common memory (CM) advantageously replaces the known two distinct memories (DM, UM) generally used, one for the downstream path and the other for the upstream path. The single common memory (CM) is particularly adapted to Very High Speed Digital Subscriber Line [VDSL-, VDSL or VDSL+] devices where the two downstream frequency ranges (DF1, DF2) are separated by an upstream frequency range (UF1); and a where a second upstream frequency range (UF2) may exist. The size of the common memory shared by the fourth circuits is slightly larger (2800 carriers of 16 bits) than one (2048 carriers of 16 bits) of the known two memories interfacing each only two circuits of a same path. However, the size of this common memory is smaller than the sum (2×2048 carriers of 16 bits) of these two distinct memories.

187. Hoogenbemt teaches using a shared memory as a replacement or improvement to two distinct memories where one memory is for downstream and one memory for upstream. This optimizes the “size of the memory means” which in turn reduces “the cost of the chip because we are using less silicon” See excerpts of ¶¶5-9 of Hoogenbemt:

**[0005]** An object of the present invention is to provide a DSL telecommunication device of the above known type but wherein the size of the memory means is optimized, thereby reducing the cost of the chip because of using less silicon.

**[0006]** According to the invention, this object is achieved due to the fact that said telecommunication device comprises

a plurality of common memories shared by circuits of said first path and by corresponding circuits of said second path, each common memory of said plurality interfacing two circuits of said first path and two corresponding circuits of said second path and being adapted to store first data transferred between said two circuits of said first path and to store second data transferred between said two corresponding circuits of said second path.

**[0007]** In this way, each pair of different memories used to latch first data of the first, e.g. the downstream, path and second data of the second, e.g. the upstream, path is replaced by a single common memory. This is possible because first data and second data are not overlapping in the frequency domain, a common memory may thus be used. This common memory is accessed both by circuits of the first path and by circuits of the second path.

**[0008]** At almost each stage of the data process in the paths, it is possible to use a memory that is common for both the second and the first data, each circuit or processor using only a part of this shared common memory.

**[0009]** Since the die or chip size is thereby even more reduced, the number of lines per chip can be increased and the production cost reduced.

188. I note that Hoogenbemt observes that at almost each stage of the data process in the paths, it is possible to replace separate memories for upstream and downstream processing with a smaller amount of memory that is part of a shared common memory. Hoogenbemt notes that the die or ship size is even further reduced when additional separate upstream and downstream memories are replaced by use of common shared memory.

189. I also note that Hoogenbemt explicitly identifies the VDSL environment as an explicit example for sharing a common memory between upstream and downstream functions.

**[0025]** Although applicable to most of the memory means of any DSL device, the following part of the specification will take as an example the second downstream memory means or memory DM and the third upstream memory means or memory UM to describe the invention in a VDSL environment.

190. In the VDSL example, the Hoogenbemt specification gives the example of sharing a common memory between the FFT circuit, the Demapper/Viterbi, the Mapper/Viterbi circuit, and the IFFT circuit.

[0029] The common memory CM is shared by the processors of the upstream and downstream paths. In the present example and as shown at **FIG. 3**, the common memory CM is for instance shared between:

[0030] the Fast-Fourier-Transform circuit FFT of which an output is connected to a first input DDI of the common memory CM;

[0031] the Demapper/Viterbi circuit of which an input is connected to a first output DDO of CM;

[0032] the Mapper/Viterbi circuit of which an output is connected to a second input UDI of CM; and

[0033] the Inverse-Fast-Fourier-Transform circuit IFFT of which an input is connected to a second output UDO of CM.

191. However, Hoogenbempt clearly and explicitly states that replacing two memories by a single common memory of smaller size is applicable for most of the memory means of any DSL device. A POSA would understand that Hoogenbempt's approach can thus be applied to replace separate memories for an upstream interleaver and a downstream deinterleaver with a single common memory of smaller size or to replace separate memories for an downstream interleaver and a upstream deinterleaver with a single common memory of smaller size.

#### **E. Configuration Messages**

192. I understand that a POSA at the time of the invention would have understood that whenever two transceivers have multiple communication configurations from which to choose, the transceivers must have a way to learn each other's capabilities and to agree on the specific configuration that they will use to communicate.

193. It is a basic principle that would have been well understood by a POSA at the time of the invention that whenever two transceivers have multiple communication configurations from which to choose, messages can be sent from one transceiver to another to establish which specific configuration will be employed. Indeed, such configuration messages are typically the

first and only approach considered to solve the problem of ensuring that the transceivers agree on which configuration is being used. Such configuration messages had been employed in numerous systems well before the date of the alleged inventions.

194. As an example, consider U.S. Patent No. 4,924,456 entitled “High Speed Modem” filed August 29, 1988 and issued May 8, 1990. This patent describes a modem that uses configuration messages to change its speed (data rate) based on how much data awaits transmission based on the contents of a transmit data buffer and also based on the quality of the telephone line based on how many retransmissions are being requested. Below is the abstract for U.S. Patent No. 4,924,456:

**[57]**

**ABSTRACT**

**A modem is disclosed having a data transmission protocol involving lower-speed, full-duplex operation during the connect sequence with a remote modem and an automatic switch to higher-speed, half-duplex operation for data transfer. Further, the modem data transmission involves transparently changing between lower-speed, interactive operation and higher-speed operation based upon data transmission demands. The operation is controlled by a processor monitoring the contents of a transmit data buffer and providing a mode control command to the modem transmitter. The modem also adapts its speed to the quality of the telephone line by fallback or fallforward to a different speed based upon predetermined data frame retransmission criteria.**

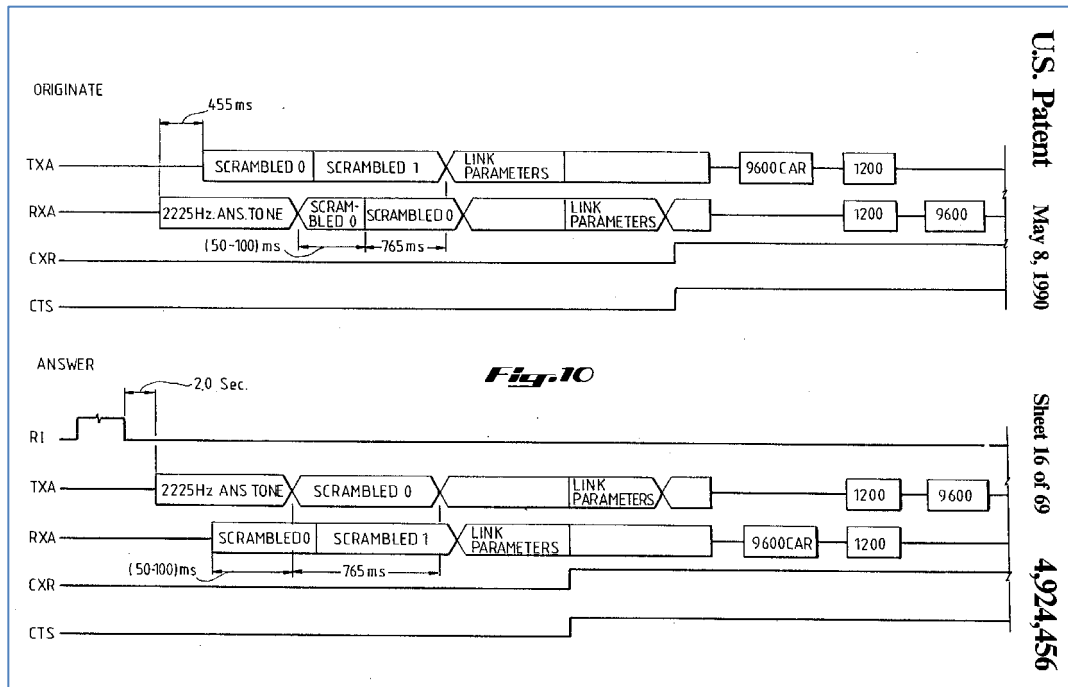
195. U.S. Patent No. 4,924,456 explicitly describes the “handshake” process which includes the exchange of configuration messages that allow for “the communication of a number of parameters” in 4:9-35 shown below. Note in particular the statement in 4:28-31 that “The various exchanges between an originating modem and an answering modem in the handshake sequence and in linking are well known to those of skill in the modem art.” This statement affirms that at least by 1988 the use of configuration messages was well known to a POSA.

10 It is to be understood that the modem also places signals  
 on the telephone line that allow the answering modem  
 to link with the originating modem. This involves com-  
 munication of a number of parameters that allow the  
 receiver to establish carrier detection, adjust automatic  
 gain control circuits, establish timing synchronization,  
 15 converge and adapt the equalizer to the initial line con-  
 ditions, and synchronize the descrambler. Also, in the  
 initial linking procedure, referred to in the art as the  
 “handshake” sequence, tones will be placed on the tele-  
 phone line.  
 20 In addition to accepting digital data from the data  
 terminal for communication, i.e., TXD, and providing  
 received digital data to the data terminal, i.e., RXD,  
 there may be an exchange between the modem and the  
 data terminal of various control signals such as data  
 25 terminal ready (DTR), request to send (RTS), data set  
 ready (DSR), clear to send (CTS) and Carrier (CXR).  
 The various exchanges between an originating modem  
 and an answering modem in the handshake sequence  
 and in linking are well known to those of skill in the  
 30 modem art. Similarly, the various exchanges between a  
 modem and the data terminal to which it is coupled  
 including the above-mentioned DTR, RTS, DSR, CXR  
 and CTS signals are also well-known to those of skill in  
 35 the modem art.

196. U.S. Patent No. 4,924,456 explains that a modem uses configuration messages in  
 an “Interactive state . . . to control its operation or to configure its operating characteristics.” 456  
 Patent at 11:30–32. Likewise, the V.32 bis, and V.32 bis modem standards explicitly handle  
 transceivers with different capabilities since a V.32 bis modem communicating with an older  
 V.32 modem will learn of the limited rate capabilities of the older modem through configuration  
 messages and choose a rate that both modems can implement.

197. Figure 10 from U.S. Patent No. 4,924,456 illustrates the connect sequence for the  
 modem described in the patent. That figure is reproduced as Figure 30 below.





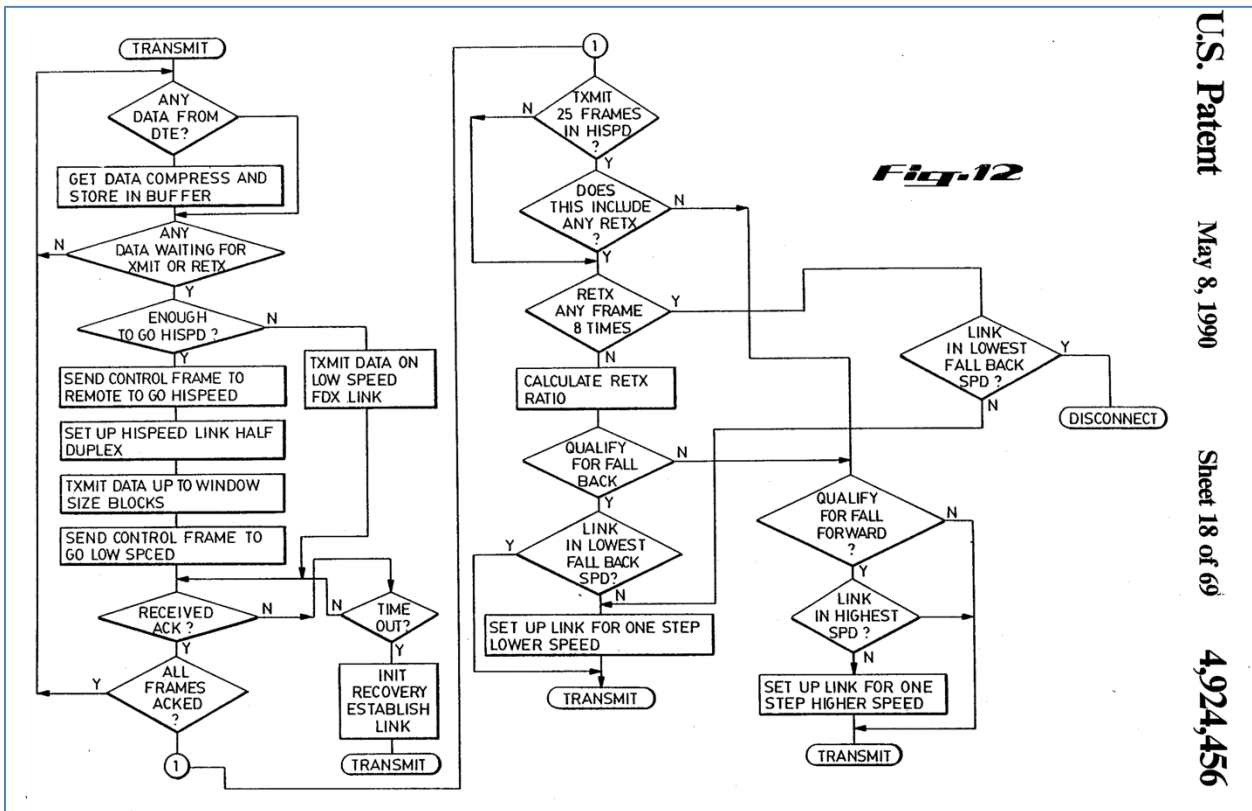
**Figure 30: Figure 10 from U.S. Patent No. 4,924,456 shows the exchange of configuration messages (link parameters) during initialization of a modem.**

198. The exchange of link parameters illustrated in Figure 30 is described in the patent at 18:33-40:

The modems then exchange link parameters for error control. The originate modem first sends approximately 23 bytes of link parameter information. The answer modem receives the link parameters and if any parameters need to be changed, it sends update link parameters to the originate modem. After the update link parameters are exchanged, the CXR and CTS lines are turned on at both modems.

199. Configuration messages are also understood by a POSA to be useful for adapting the communications configuration to specific channel conditions and in consideration of available memory requirements. As described in U.S. Patent No. 4,924,456, configuration messages can be used after initialization to change the configuration adaptively. An example of this is provided by Figure 12 of U.S. Patent No. 4,924,456, which is shown below as **Error! Reference source not found..** Note in the figure the blocks that indicate configuration messages: “SEND CONTROL FRAME TO REMOTE TO GO HISPEED,” “SEND CONTROL FRAME TO GO LOW

SPEED,” “SET UP LINK FOR ONE STEP LOWER SPEED,” and “SET UP LINK FOR ONE STEP HIGHER SPEED.”



**Figure 31: Figure 12 from U.S. Patent No. 4,924,456 shows the exchange of configuration messages (control frames) that change the configuration to high speed and then back to low speed based on the amount of user data stored in the buffer. The figure also shows adjusting the data rate dynamically based on the number of retransmissions and the retransmission ratio.**

200. The use of configuration messages to adaptively change the configuration as illustrated in Figure 12 of U.S. Patent No. 4,924,456 “automatically changes between speeds of transmission based upon data transmission demand during a communication and upon the quality of the data transmissions.” (See 19:31-34). As shown in the above figure, these automatic changes in transmission speed are initiated by configuration messages from the modem and the remote modem to which it is transferring data. The above figure is described explicitly in lines 19:30-20:25 of U.S. Patent No. 4,924,456, which are excerpted below.

FIG. 12 flowcharts an operational sequence of the 30  
modem involving data communication that automati-  
cally changes between speeds of transmission based  
upon data transmission demand during a communica-  
tion and upon the quality of the data transmissions.  
When the modem begins transmitting, it checks for data 35  
from the data terminal. The data for transmission is  
compressed and stored in the transmit data buffer. Next,  
there is evaluation of whether any data is waiting for  
retransmission. If so and there is enough to go higher-  
speed, the modem sends control frame information to 40  
the remote modem to go higher-speed. The higher-  
speed, half-duplex link is set up and transmission of data  
begins. At the conclusion of data transmission, a control  
frame is sent requesting the remote modem to go into  
the lower-speed mode. The modem then awaits recep- 45  
tion of an acknowledgement. If the acknowledgement  
does not come within a prescribed time out period, the  
modem initiates recovery by establishing the link and  
reentering the transmit sequence. If the request to go  
lower-speed is acknowledged, the routine goes back to 50  
the beginning of the transmit sequence.

If there is no acknowledgement that all data frames  
were received properly, there is consideration of  
whether a predetermined number of data frames have  
been transmitted at higher-speed (e.g. 25 frames). If so, 55  
there is a check to determine whether the transmitted  
data frames included any retransmissions. If there was  
retransmission of any data frame for predetermined  
number of times (e.g. 8 times), there is a calculation of  
the retransmit frame ratio. Based upon the calculated 60  
ratio, a determination is made whether the quality of the  
data transmission indicates qualification for a fallback in  
speed. If there had been retransmission of any data  
frame at least the predetermined number of times, and if  
the modem is linked in the lowest fallback speed, then 65  
the modem disconnects. If the link is not in the lowest  
fallback speed, the modem sets up the link for one step  
lower in speed and reenters the transmit sequence.

24,456

20

If the evaluation to determine whether a predeter-  
mined number of preceding transmitted frames included  
any retransmitted data indicates that none were retrans-  
mitted frames, the routine makes an evaluation for qual-  
ification to fallforward to a higher speed. If the qualifi-  
cation is not met, the transmit sequence is reentered at  
the same speed. But if the data transmission quality  
qualifies for a fallforward in speed, and the link is not  
already in the highest speed, the modem sets up the link  
10 for a one step higher speed of transmission.

The routine in FIG. 12 illustrates the ability of the  
modem to adapt its speed to the quality of the telephone  
line. As indicated, the modem can either fallback to a  
lower speed or fallforward to a higher speed. In gen-  
eral, to determine the quality of the line, the processor  
15 of the modem constantly monitors the number of errors  
in data transmission as reflected by the number of re-  
transmitted data frames. If the number of retransmis-  
sions is high, indicating too many errors are encoun-  
tered and line quality is poor, the modem drops down to  
20 the next lower speed until an acceptable reduction in  
errors in transmission is achieved. If the line quality  
improves and the number of errors is reduced, the  
modem will automatically fallforward to the next  
25 higher speed.

201. Specific quotes from lines 19:30-20:25 of the '456 patent that describe  
configuration messages include "If so and there is enough to go higher-speed, the modem sends  
control frame information to the remote modem to go higher-speed." (19:39-41.), "At the  
conclusion of data transmission, a control frame is sent requesting the remote modem to go into  
the lower-speed mode." (19:43-45), "If the link is not in the lowest fallback speed, the modem  
sets up the link for one step lower in speed and reenters the transmit sequence." (19:66-68), and  
"But if the data transmission quality qualifies for a fallforward in speed, and the link is not  
already in the highest speed, the modem sets up the link for a one step higher speed of  
transmission. (20:7-10).

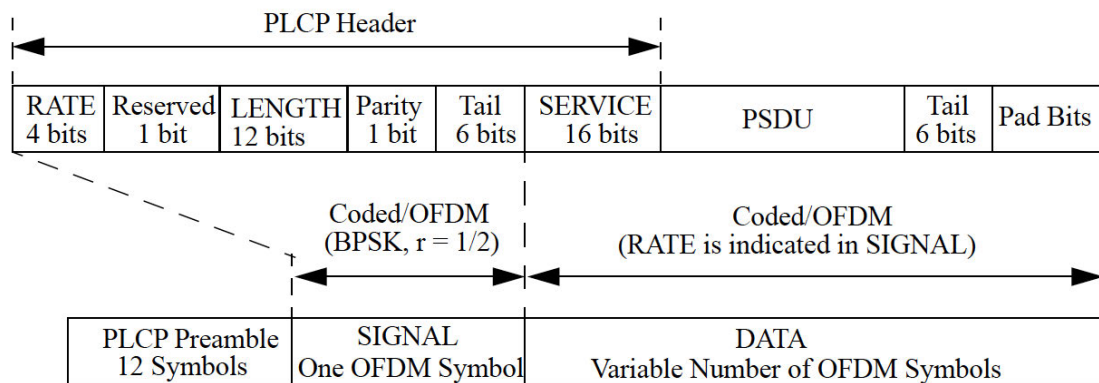
202. Note that the cases described above in Figure 12 of the '456 patent and the accompanying text, these configuration changes are intimately related to how much memory is being utilized by certain buffers. The “data transmission demand” is determined by how much data is waiting in the transmit data buffer. The number of times a frame has been retransmitted and the retransmission ratio are both closely related to how much memory is required by the retransmission buffer. For example, in 8:37-39 of the '456 patent, the dependence of modem speed and memory pressure on the transmit data buffer is explicitly stated: “If the modem can keep its transmit data buffer from filling using the lower-speed mode, it will not go to high speed.” As another example, the choice of when to change speeds based on the retransmission behavior would have been understood by a POSA to be related to the number of unacknowledged packets (outstanding packets) that can be accommodated by the retransmission buffer. This memory allocation is discussed, for example, at 11:26-29: “The Error Control Mode provides for synchronous data packets operation and enables selective ARQ, dynamic fallback/fallforward, and data compression with an outstanding packet window of 12.”

203. Configuration messages in a multicarrier transmission system with multicast capability are found, for example, in 802.11a as described in the IEEE Std 802.11a-1999, which added OFDM capability to the 1997 802.11 standard. Sections 17.3.2 and 17.3.4, reproduced in part below, describe the physical layer convergence protocol (PLCP) frame format of the PLCP protocol data unit (PPDU), which includes configuration messages in the OFDM PLCP header also referred to in Figure 107 of 802.11a as the signal field (SIGNAL). The signal field provides configuration information required for successful communication including the length of the transmission, the coding rate, and the type of modulation to be used in the rest of the packet.



### 17.3.2 PLCP frame format

Figure 107 shows the format for the PPDU including the OFDM PLCP preamble, OFDM PLCP header, PSDU, tail bits, and pad bits. The PLCP header contains the following fields: LENGTH, RATE, a reserved bit, an even parity bit, and the SERVICE field. In terms of modulation, the LENGTH, RATE, reserved bit, and parity bit (with 6 “zero” tail bits appended) constitute a separate single OFDM symbol, denoted SIGNAL, which is transmitted with the most robust combination of BPSK modulation and a coding rate of  $R = 1/2$ . The SERVICE field of the PLCP header and the PSDU (with 6 “zero” tail bits and pad bits appended), denoted as DATA, are transmitted at the data rate described in the RATE field and may constitute multiple OFDM symbols. The tail bits in the SIGNAL symbol enable decoding of the RATE and LENGTH fields immediately after the reception of the tail bits. The RATE and LENGTH are required for decoding the DATA part of the packet. In addition, the CCA mechanism can be augmented by predicting the duration of the packet from the contents of the RATE and LENGTH fields, even if the data rate is not supported by the station. Each of these fields is described in detail in 17.3.3, 17.3.4, and 17.3.5.

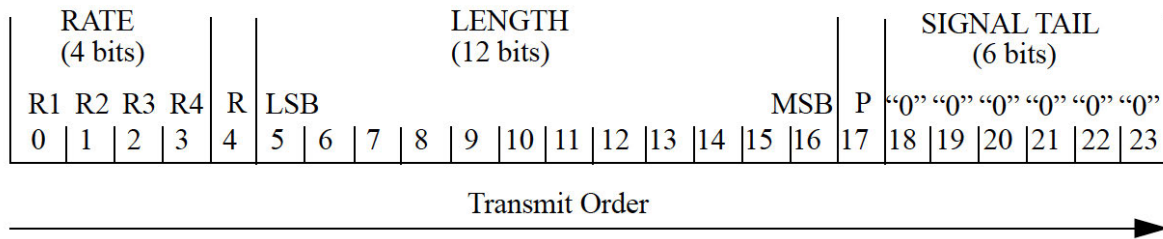


**Figure 107—PPDU frame format**

### 17.3.4 Signal field (SIGNAL)

The OFDM training symbols shall be followed by the SIGNAL field, which contains the RATE and the LENGTH fields of the TXVECTOR. The RATE field conveys information about the type of modulation and the coding rate as used in the rest of the packet. The encoding of the SIGNAL single OFDM symbol shall be performed with BPSK modulation of the subcarriers and using convolutional coding at  $R = 1/2$ . The encoding procedure, which includes convolutional encoding, interleaving, modulation mapping processes, pilot insertion, and OFDM modulation, follows the steps described in 17.3.5.5, 17.3.5.6, and 17.3.5.8, as used for transmission of data at a 6 Mbit/s rate. The contents of the SIGNAL field are not scrambled.

The SIGNAL field shall be composed of 24 bits, as illustrated in Figure 111. The four bits 0 to 3 shall encode the RATE. Bit 4 shall be reserved for future use. Bits 5–16 shall encode the LENGTH field of the TXVECTOR, with the least significant bit (LSB) being transmitted first.

**Figure 111—SIGNAL field bit assignment**

The process of generating the SIGNAL OFDM symbol is illustrated in Annex G (G.4).

#### 17.3.4.1 Data rate (RATE)

The bits R1–R4 shall be set, dependent on RATE, according to the values in Table 80.

**Table 80—Contents of the SIGNAL field**

Rate (Mbits/s)	R1–R4
6	1101
9	1111
12	0101
18	0111
24	1001
36	1011
48	0001
54	0011

**17.3.4.2 PLCP length field (LENGTH)**

The PLCP length field shall be an unsigned 12-bit integer that indicates the number of octets in the PSDU that the MAC is currently requesting the PHY to transmit. This value is used by the PHY to determine the number of octet transfers that will occur between the MAC and the PHY after receiving a request to start transmission. The transmitted value shall be determined from the LENGTH parameter in the TXVECTOR issued with the PHY-TXSTART.request primitive described in 12.3.5.4 (IEEE Std 802.11, 1999 Edition). The LSB shall be transmitted first in time. This field shall be encoded by the convolutional encoder described in 17.3.5.5.

**17.3.4.3 Parity (P), Reserved (R), and Signal tail (SIGNAL TAIL)**

Bit 4 shall be reserved for future use. Bit 17 shall be a positive parity (even parity) bit for bits 0–16. The bits 18–23 constitute the SIGNAL TAIL field, and all 6 bits shall be set to zero.

204. Another example of configuration messages is provided in U.S. Patent No. 7,027,782 (“Moon”), which was filed in October of 2002. Moon at 15:41-51. Moon teaches that the receiver has to understand each operation performed by the transmitter so that the receiver can perform the corresponding operation.

Meanwhile, the receiver should previously recognize information on the coding rate, the modulation technique, the orthogonal codes, and the number of orthogonal codes, used by the transmitter illustrated in FIG. 7, and the number of retransmissions, for demodulation and decoding. That is, the above information should be previously provided to the despreader 812, the demodulator 814, the selective packet combiner 816, and the decoder 824 so that the receiver can perform a corresponding operation of the transmitter. Therefore, the above information is provided from the transmitter to the receiver over a downlink control channel.

(Moon 15:41-51)

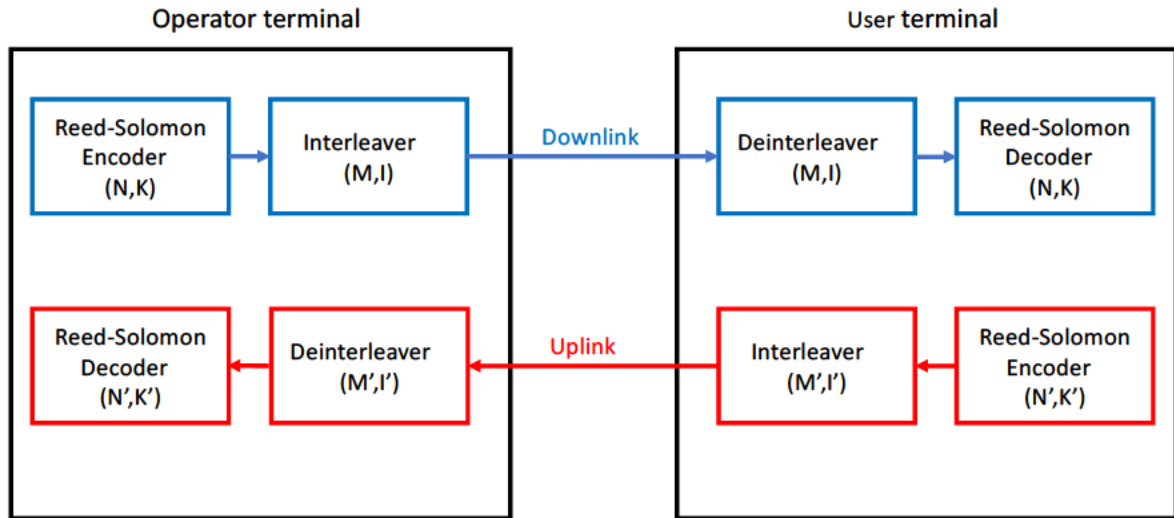
205. Information that the transmitter and receiver exchange to agree on how they will communicate affects the allocation of shared memory between interleaving and retransmission in the system described by Moon. For example, depending on the coding rate applied, the relative inputs to the first and second interleaver memories can change. With a coding rate of  $\frac{1}{2}$ , the first and second interleaver memories will receive a symmetric input, i.e., for each bit that goes into the



first interleaver, one bit will go into the second interleaver. Moon at 12:12-14. But for a coding rate of  $\frac{3}{4}$ , for every three bits that go into the first interleaver, only one bit will go into the second interleaver. Moon at 12:15-18. Similarly, the choice of modulation technique will affect the amount of memory that must be allocated for retransmission.

206. Another example of configuration messages is the messages that would be used in the system disclosed by Mazzoni, messages between the two transceivers that would allow the system to adapt the sharing of memory between interleaving and deinterleaving. As disclosed in Mazzoni, “Still another object of the invention is to provide such an architecture which is adaptable, particularly in terms of the memory capacity of the interleaving and deinterleaving means, to suit a number of different bit rates selected from a predetermined group of bit rates.” Mazzoni at 1:54-58. The two transceivers must exchange information about their capabilities and the channel characteristics to select the most appropriate rate from a predetermined group of rates.

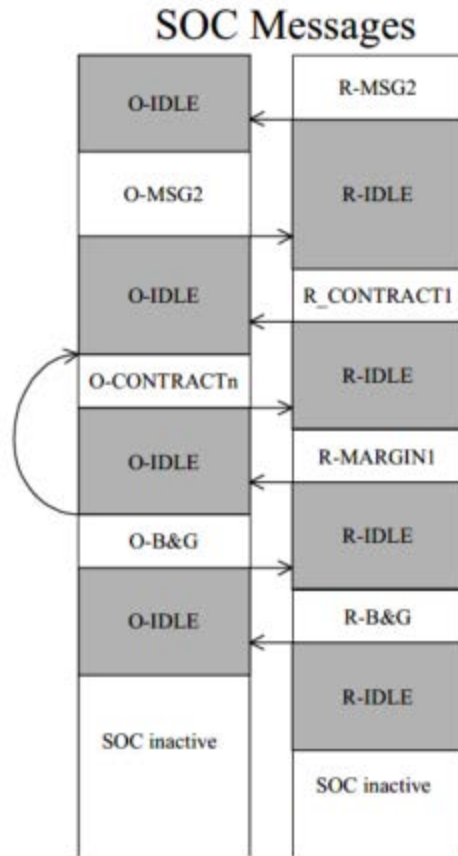
207. Consider the system below with a transceiver at an operator terminal communicating with a transceiver at a user terminal. The communication channel from the operator to the user is called the downlink, and the channel from the user to the operator is called the uplink. In this example system, as in many prior art systems, including ADSL and VDSL1, Reed-Solomon coding and interleaving are used for both the uplink channel and the downlink channel. A POSA would have understood that for the communication link to function properly, the Reed-Solomon decoder and convolutional deinterleaver in the user transceiver for the downlink must use the exact same parameters  $(N,K,M,I)$  as the Reed-Solomon encoder and interleaver for the downlink in the operator terminal. Similarly, the operator and user must agree on parameters  $(N',K',M',I')$  for the Reed-Solomon coding and interleaving for the uplink.



208. A POSA would have known that a common solution is to use configuration messages to exchange capabilities and agree on parameters. Such configuration messages are used in ADSL standards as well as in the VDSL1 standard.

209. Additionally the ETSI TS 101 270-2 VDSL standard (ETSI VDSL) (NOK00076413), which was published on July 24, 2003, discloses communicating the capabilities of the transceiver in a configuration message, including by delivering a message specifying the maximum available interleaver memory. See, e.g., ETSI VDSL at 132.

210. ETSI VDSL teaches initialization protocols between the VTU-O and VTU-R to set interleaver parameters, including the messages shown below. *Id.* at 127.



211. The first message, R-MSG2, describes the VTU-R's capabilities, including the maximal number of bytes available to be allocated to an interleaver. *Id.* at 132. After the messages R-MSG2, O-MSG2, and R-CONTRACT1, the message O-CONTRACTn, contains a proposed upstream contract and a proposed downstream contract, including contract descriptors for each contract, including the parameters I and M for the upstream interleaver/deinterleaver and a distinct set of parameters I and M for the downstream deinterleaver/interleaver. *Id.* at 129.

212. Figure 50, shown below, shows the "Channel analysis and exchange" state as part of initialization. *Id.*

VTU-O		
Activation: Handshake procedures (clause 8.2.3)	Training (clause 8.2.4)	Channel analysis and Exchange (clause 8.2.6)
VTU-R		
Activation: Handshake procedures (clause 8.2.3)	Training (clause 8.2.4)	Channel analysis and Exchange (clause 8.2.6)

**Figure 50: Overview of initialisation**

213. A POSA would have understood that the description of Figure 50 discloses that during the channel analysis and exchange state, the VTU-O and VTU-R measure the characteristics of the channel and agree on all the parameters needed to thoroughly define the communication link. Specifically, ETSI VDSL discloses:

The time line in Figure 50 provides an overview of the initialisation protocol. Following the initial handshake procedure, a full duplex link between the VTU-O and the VTU-R is established. During the training phase, timing advance and upstream power back-off shall be refined. During the channel analysis and exchange state, the two modems measure the characteristics of the channel and agree on a contract that thoroughly defines the communication link. *Id.* 111

214. During the “Channel analysis and exchange” state, the VTU-R sends a specific message, R-MSG2, to the VTU-O to describe its capabilities, including its maximum available interleaver memory. *Id.* at 132. For example, the message R-MSG2 includes the field “Maximal interleaver memory,” which specifies the maximal number of bytes available to be allocated to an interleaver. *Id.* at 132. The R-MSG2 contains information about the capabilities of the VTU-R for bit allocation. *Id.* The content of R-MSG2 is specified in Table 82, which is shown below.

**Table 82: Description of R-MSG2**

Field	Field or macro-field type	Remark
Message descriptor	1 octet	See table 49
Maximum constellation size in upstream	1 octet	Maximum number of bits per tone
RS setting supported by VTU-R	1 octet	0x00: only mandatory settings 0xFF: all settings (see note 1)
Interleaver setting supported by VTU-R	1 octet	0x00: only mandatory settings 0xFF: all settings 0xNN: NN = number of settings (0x00 < NN < 0xFF)
Detailed interleaver setting description	0 octets if NN = 0x00 0 octets if NN = 0xFF NN x 4 octets otherwise	See table 74
Maximum power transmitted	1 octet	In units of 0,25 dBm
Maximum interleaver memory	3 octets	In octets (see note 2)
Maximum number of EOC octets per frame in upstream	1 octet	Number of EOC octets per frame
Maximum number of VOC octets per frame in upstream	1 octet	Number of VOC octets per frame
Support of express swapping	1 octet	0x00: Not supported 0xFF: Supported
$j_{\max}$	1 octet	Maximum value of $j_{\max}$ supported by the VTU-R (see note 3)
NOTE 1: All settings means the values (for the redundancy) that are specified in clause 6.4.2.		
NOTE 2: The interleaver memory is computed as $M \times l \times (l-1)$ .		
NOTE 3: Specification of $j_{\max} = k$ means that all values from 0 to k are supported.		

215. After receiving this message, the VTU-O sends the message O-MSG2 describing its capabilities. *Id.* at 127. After receiving the O-MSG2, the VTU-R sends message R-CONTRACT1 to the VTU-O to propose parameters for the downstream channel Reed-Solomon coding and interleaving, within the capabilities as set forth in the R-MSG2 and O-MSG2 messages. *Id.* at 132. After receiving R-CONTRACT1, the VTU-O responds with its proposed parameters for both upstream and downstream channels using the O-CONTRACT1 message. *Id.* at 129. The O-CONTRACT1 message bases the downstream contract on the interleaving parameters proposed in R-CONTRACT1. Ideally the downstream contract is the same as the one proposed in R-CONTRACT1. *Id.* at 129. A POSA would have understood that the upstream contract proposed by the VTU-O must comport with the limitations of the VTU-R including maximum interleaver memory as communicated in R-MSG2.

216. When the VTU-R receives the O-CONTRACT1 message, it computes the minimal SNR margin over all tones and sends this information to the VTU-O in the R-MARGIN1 message. *Id.* at 132. If the VTU-O finds the SNR margins in the R-MARGIN1 message acceptable, then it sends the O-B&G message, which ends the negotiation. *Id.* at 129–30. If the VTU-O does not find the SNR margins in the R-MARGIN1 message acceptable, then it sends a new O-CONTRACT2 message and the VTU-R computes the new values for the minimal SNR margin over all tones and sends this information to the VTU-O in the R-MARGIN2 message. *Id.* at 129–32. The VTU-O and VTU-R continue to exchange O-CONTRACT<sub>n</sub> and the R-MARGIN<sub>n</sub> messages until the VTU-O is satisfied and sends the O-B&G message which ends the negotiation. *Id.*

217. A POSA would have understood that, throughout the exchange, the VTU-O would not propose a contract that exceeds the available interleaver memory at the VTU-R, which is described in the “maximum interleaver memory” field of the R-MSG2 sent by the VTU-R.

218. Thus, fifteen years before the alleged invention, configuration messages were considered well-known, e.g., “The various exchanges between an originating modem and an answering modem in the handshake sequence and in linking are well known to those of skill in the modem art.” (U.S. Patent No. 4,924,456 at 4:28-31). Configuration messages that allow two transceivers to exchange messages that communicate their capabilities and to arrive at agreed-upon parameters that “thoroughly define the communication link” had been in use for at least that long. Such configuration messages are used by all ADSL standards and the VDSL standards such as ETSI VDSL and VDSL1 that were available before the alleged invention. Certainly by July of 2003, a message communicating the maximum number of bytes of memory that are available to be allocated to an interleaver had been specified in an international VDSL standard, e.g. the Maximum interleaver memory field of R-MSG2 in ETSI VDSL.

219. In June of 2004, Texas Instruments, Inc. proposed LB-031, an ITU-T contribution that came after the ETSI VDSL standard and acknowledges that one known option for communicating interleaver capabilities was to specify in a message “an amount of memory.” See LB-031 at 6.

220. LB-031 recognized that different interleaver implementations use different amounts of memory and the actual amount of memory is implementation specific, See below:

The smallest amount of memory required to build an interleaver/deinterleaver pair is equal to the total delay of the interleaver/deinterleaver [3]. Typically, for memory optimized interleavers, the interleaver and deinterleaver memory size is nearly the same. Therefore, the smallest possible memory for either the interleaver or deinterleaver is

$$\text{smallest possible (de)interleaver memory} = (I - 1) \cdot (d - 1) / 2 \text{ bytes.} \quad (2)$$

In a typical implementation, slightly more memory is often required. The actual amount of required memory is implementation specific.

(LB-031 at 2)

221. Unless a standard chooses to force all manufacturers to adopt the same implementation, memory is not precise way to communicate which interleavers a transceiver can support. LB-031 suggested that interleaver complexity should be specified in terms of delay and not in terms of memory or an interleaver depth. According to LB-031, interleaver delay (which is not an amount of memory) should be exchanged in terms of octets to indicate the interleaving capabilities of transceivers. A POSA would have understood that delay and an amount of memory are distinct concepts as indicated in LB-031 because an interleaver with a specified delay can be implemented using various amounts of memory depending on the choice of implementation. This was described above in Section **Error! Reference source not found.** Through LB-031, Texas Instruments therefore proposed specifying interleaver complexity in terms of delay and not in terms of an amount of memory as an improvement over what was known.



## 6. Proposal

This contribution addresses

11.4	Open	What shall be the mandatory interleaver capabilities?	MC-086, D1060
------	------	---	---------------

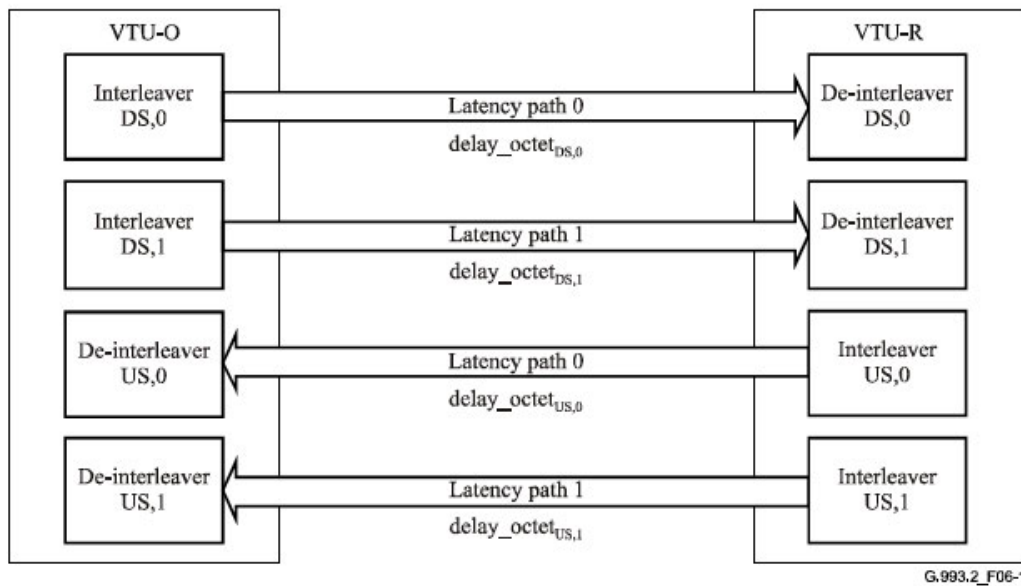
and proposes:

- interleaver complexity should be specified in terms of a time delay (ms), not in terms of an amount of memory or an interleaver depth
- interleaver delay in terms of octets should be exchanged between the VTU-O and VTU-R; the delay in octets should meet the minimum requirements in terms of the delay in time
- the upper limit on the number of codewords per unit time should be constrained and should scale with the data rate so that as the data rates increase, this upper limit on the number of codewords is also higher

222. This idea of specifying interleaver complexity by indicating a maximum delay in terms of octets and not in terms of an amount of memory was ultimately adopted in the VDSL2 standard ITU-T G.993.2, as shown below. G.993.2 indicates communicating interleaver complexity as delay in terms of octets and explicitly states that the actual amount of memory used is implementation specific and thus not communicated by the delay. (VDSL2 page 22-23)

### 6.2.8 Aggregate interleaver and de-interleaver delay

The required aggregate interleaver and de-interleaver delay is specified in terms of the sum of the end-to-end delays in the upstream and downstream directions over both latency paths, expressed in octets. Therefore, it involves both VTUs. Figure 6-1 illustrates an end-to-end connection with two latency paths and their interleavers and de-interleavers.



**Figure 6-1/G.993.2 – Illustration of all latency paths composing the aggregate interleaver and de-interleaver delay specified in each profile**

The end-to-end delay in octets for the interleaver and de-interleaver pair on path  $p$ , with  $p = 0, 1$ , is given by:

$$\text{delay\_octet}_{x,p} = (I_{x,p} - 1) \times (D_{x,p} - 1)$$

where the direction of transmission  $x$  is either "DS" for downstream or "US" for upstream,  $I_{x,p}$  is the interleaver block length, and  $D_{x,p}$  is the interleaver depth.

Each interleaver and each de-interleaver for each latency path requires at least  $(\text{delay\_octet}_{x,p}/2)$  octets of memory to meet this delay. The actual amount of memory used is implementation specific.

Referring to Figure 6-1, the aggregate interleaver and de-interleaver delay is specified as the sum  $\text{delay\_octet}_{\text{DS},0} + \text{delay\_octet}_{\text{DS},1} + \text{delay\_octet}_{\text{US},0} + \text{delay\_octet}_{\text{US},1}$ ,

which can be rewritten as:

$$\sum_p (I_{\text{US},p} - 1) \cdot (D_{\text{US},p} - 1) + (I_{\text{DS},p} - 1) \cdot (D_{\text{DS},p} - 1)$$

VDSL2 modems shall comply with the requirement

$$\sum_p (I_{\text{US},p} - 1) \cdot (D_{\text{US},p} - 1) + (I_{\text{DS},p} - 1) \cdot (D_{\text{DS},p} - 1) \leq \text{MAXDELAYOCTET},$$

where the summation is over all latency paths and MAXDELAYOCTET is the parameter "aggregate interleaver and de-interleaver delay", in octets, specified in Table 6-1 for the profile.

The minimum amount of memory required in a transceiver (VTU-O or VTU-R) to meet this requirement is  $\frac{\text{MAXDELAYOCTET}}{2}$  octets. The actual amount of memory used is implementation specific.

223. In VDSL2, the O-PMS message is used to communicate maximum interleaver delay in terms of octets. The actual amount of memory used in each transceiver is implementation specific according to VDSL2. Communicating interleaver delay instead of an amount of memory allowed transceivers to precisely specify what interleavers they could support without worrying about implementation details that are not usually addressed in standards specifications. ETSI VDSL and VDSL1 both specify a specific triangular implementation for convolutional interleaving and communicate a maximum amount of memory available for interleaving in R-MSG2. In contrast, the later in time VDSL2 standard does not discuss the specific interleaver implementation and communicates interleaver complexity in terms of delay rather than memory. It was thus well known, at least as of June 2004, that using interleaver delay was a configuration scheme that could be used and implemented in VDSL.

## **IX. THE FAMILY 3 PATENTS**

### **A. Overview of the Family 3 Patents**

224. The Family 3 Patents relate to sharing a common memory and allocating that common memory between an interleaver and a deinterleaver in digital subscriber line (“DSL”) technology. ’882 Patent at Abstract.

225. I have been informed by counsel that plaintiff TQ Delta is asserting claims 9 and 13 of the ’882 Patent, claim 5 of the ’048 Patent, claim 28 of the ’5473 Patent, and claim 2 of the ’608 Patent.

226. I understand that TQ Delta claims that the alleged inventions in all of the asserted claims in each asserted patent are disclosed in an October 12, 2004 Provisional Application (hereinafter the “October 2004 Provisional”). As I noted above, I disagree that the asserted claims of the Family 3 Patents are entitled to the priority date of the October 2004 Provisional.

## 6. Proposal

This contribution addresses

11.4	Open	What shall be the mandatory interleaver capabilities?	MC-086, D1060
------	------	---	---------------

and proposes:

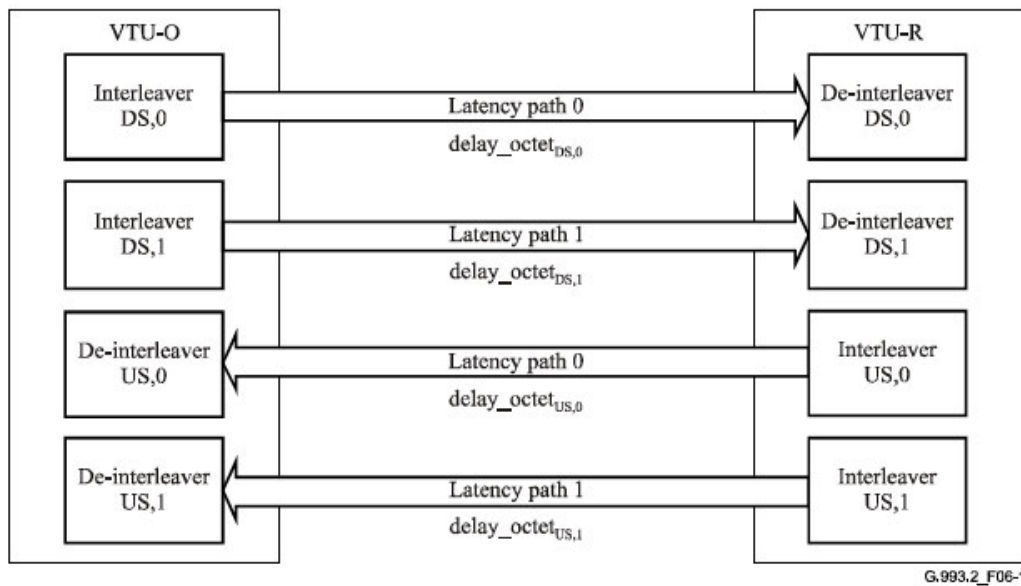
- interleaver complexity should be specified in terms of a time delay (ms), not in terms of an amount of memory or an interleaver depth
- interleaver delay in terms of octets should be exchanged between the VTU-O and VTU-R; the delay in octets should meet the minimum requirements in terms of the delay in time
- the upper limit on the number of codewords per unit time should be constrained and should scale with the data rate so that as the data rates increase, this upper limit on the number of codewords is also higher

222. This idea of specifying interleaver complexity by indicating a maximum delay in terms of octets and not in terms of an amount of memory was ultimately adopted in the VDSL2 standard ITU-T G.993.2, as shown below. G.993.2 indicates communicating interleaver complexity as delay in terms of octets and explicitly states that the actual amount of memory used is implementation specific and thus not communicated by the delay. (VDSL2 page 22-23)

### 6.2.8 Aggregate interleaver and de-interleaver delay

The required aggregate interleaver and de-interleaver delay is specified in terms of the sum of the end-to-end delays in the upstream and downstream directions over both latency paths, expressed in octets. Therefore, it involves both VTUs. Figure 6-1 illustrates an end-to-end connection with two latency paths and their interleavers and de-interleavers.





**Figure 6-1/G.993.2 – Illustration of all latency paths composing the aggregate interleaver and de-interleaver delay specified in each profile**

The end-to-end delay in octets for the interleaver and de-interleaver pair on path  $p$ , with  $p = 0, 1$ , is given by:

$$\text{delay\_octet}_{x,p} = (I_{x,p} - 1) \times (D_{x,p} - 1)$$

where the direction of transmission  $x$  is either "DS" for downstream or "US" for upstream,  $I_{x,p}$  is the interleaver block length, and  $D_{x,p}$  is the interleaver depth.

Each interleaver and each de-interleaver for each latency path requires at least  $(\text{delay\_octet}_{x,p}/2)$  octets of memory to meet this delay. The actual amount of memory used is implementation specific.

Referring to Figure 6-1, the aggregate interleaver and de-interleaver delay is specified as the sum  $\text{delay\_octet}_{\text{DS},0} + \text{delay\_octet}_{\text{DS},1} + \text{delay\_octet}_{\text{US},0} + \text{delay\_octet}_{\text{US},1}$ ,

which can be rewritten as:

$$\sum_p (I_{\text{US},p} - 1) \cdot (D_{\text{US},p} - 1) + (I_{\text{DS},p} - 1) \cdot (D_{\text{DS},p} - 1)$$

VDSL2 modems shall comply with the requirement

$$\sum_p (I_{\text{US},p} - 1) \cdot (D_{\text{US},p} - 1) + (I_{\text{DS},p} - 1) \cdot (D_{\text{DS},p} - 1) \leq \text{MAXDELAYOCTET},$$

where the summation is over all latency paths and MAXDELAYOCTET is the parameter "aggregate interleaver and de-interleaver delay", in octets, specified in Table 6-1 for the profile.

The minimum amount of memory required in a transceiver (VTU-O or VTU-R) to meet this requirement is  $\frac{\text{MAXDELAYOCTET}}{2}$  octets. The actual amount of memory used is implementation specific.

223. In VDSL2, the O-PMS message is used to communicate maximum interleaver delay in terms of octets. The actual amount of memory used in each transceiver is implementation specific according to VDSL2. Communicating interleaver delay instead of an amount of memory allowed transceivers to precisely specify what interleavers they could support without worrying about implementation details that are not usually addressed in standards specifications. ETSI VDSL and VDSL1 both specify a specific triangular implementation for convolutional interleaving and communicate a maximum amount of memory available for interleaving in R-MSG2. In contrast, the later in time VDSL2 standard does not discuss the specific interleaver implementation and communicates interleaver complexity in terms of delay rather than memory. It was thus well known, at least as of June 2004, that using interleaver delay was a configuration scheme that could be used and implemented in VDSL.

## **IX. THE FAMILY 3 PATENTS**

### **A. Overview of the Family 3 Patents**

224. The Family 3 Patents relate to sharing a common memory and allocating that common memory between an interleaver and a deinterleaver in digital subscriber line (“DSL”) technology. ’882 Patent at Abstract.

225. I have been informed by counsel that plaintiff TQ Delta is asserting claims 9 and 13 of the ’882 Patent, claim 5 of the ’048 Patent, claim 28 of the ’5473 Patent, and claim 2 of the ’608 Patent.

226. I understand that TQ Delta claims that the alleged inventions in all of the asserted claims in each asserted patent are disclosed in an October 12, 2004 Provisional Application (hereinafter the “October 2004 Provisional”). As I noted above, I disagree that the asserted claims of the Family 3 Patents are entitled to the priority date of the October 2004 Provisional.

227. The October 2004 Provisional describes a system in which memory or processing power can be shared between multiple FCI blocks (which the provisional states are latency paths). The October 2004 Provisional further explains that during initialization, the modems exchange information relating to the total/shared memory capability for all FCI blocks. The October 2004 Provisional notes that this information is exchanged prior to determining the configuration of the FCI blocks. Under the Detailed Description, the October 2004 Provisional provides three examples of the memory requirements for different FCI blocks.

228. The October 2004 Provisional then provides further description of the messaging aspects of its alleged invention. The October 2004 Provisional describes different information that may be transmitted, and it specifically notes that the "receiver must know how [sic] the total shared memory for all transmitter FCI blocks." The October 2004 Provisional then describes a message that communicates both the total available memory for each FCI block, and separately communicates the total available shared memory for all FCI blocks.

229. The October 2004 Provisional then confirms that exchanging a message that includes the total shared memory for all transmit or receive FCI blocks is not an exemplary embodiment, but an essential aspect of the alleged invention:

It is apparent that it is necessary for the receiver to have knowledge of the total shared memory prior to configuring the FCI blocks to meet application requirements. It is therefore an essential aspect of this invention that prior to configuring the FCI blocks the transmitting modem must send a message to the receiving modem containing information describing the total/shared memory of the transmit FCI blocks.

230. The October 2004 Provisional is clear that it is "necessary" for the receiving modem that will determine the transmission parameters for the FCI blocks to have knowledge of the total shared memory. Thus, it is an "essential aspect" of the alleged invention that the modems "must" exchange a message describing the total shared memory for all transmit FCI blocks.



231. None of the asserted claims of the Family 3 Patents include these “essential” and “necessary” aspects of the alleged invention.

232. The asserted claims of the ’882 Patent, the ’048 Patent, the ’5473 Patent, and the ’608 Patent do not include these “essential” and “necessary” aspects. Instead, as shown below, each claims transmitting or receiving a message. The message claimed by each is specific to a deinterleaver or an interleaver of a single latency path (or FCI block in the terms of the October 2004 Provisional). And, as I noted above, the October 2004 Provisional differentiates communicating memory available to a single FCI block and communicating the total memory for all FCI blocks. Because exchange of a message communicating the total amount of shared memory available is a “necessary” and “essential” aspect of the alleged invention described in the October 2004 Provisional, a POSA would not understand the inventors to be in possession of embodiments that lack this “necessary” and “essential” aspect. That being the case, the asserted claims of the ’882 Patent, the ’048 Patent, the ’5473 Patent, and the ’608 Patent, all of which lack this “necessary” and “essential” aspect, do not have sufficient written description under 35 U.S.C. § 112 in the October 2004 Provisional.

**B. File History of the Family 3 Patents**

233. Each of the Family 3 Patents claims priority through application no. 11/246,163, which issued as U.S. Patent No. 7,831,890, and which claims priority to Provisional App. No. 60/618,269.

234. In addition to the file history of each of the Asserted Patents, which I describe below, I also reviewed the prosecution history of the ’890 patent.

235. During prosecution of the ’890 Patent, all originally-filed claims of the 163 application were rejected as anticipated by U.S. Patent No. 6,707,822 to Fadavi-Ardekani. Among other limitations, the examiner found that Fadavi-Ardekani disclosed sharing a memory in a

CONTRACT1. A POSA would have understood that the upstream contract proposed by the VTU-O must comport with the limitations of the VTU-R including not exceeding the maximum interleaver memory as communicated in R-MSG2.

575. When the VTU-R receives the O-CONTRACT1 message, it computes the minimal SNR margin over all tones and sends this information to the VTU-O in the R-MARGIN1 message. If the VTU-O finds the SNR margins in the R-MARGIN1 message acceptable, then it sends the O-B&G message, which ends the negotiation. If the VTU-O does not find the SNR margins in the R-MARGIN1 message acceptable, then it sends a new O-CONTRACT2 message and the VTU-R computes the new values for the minimal SNR margin over all tones and sends this information to the VTU-O in the R-MARGIN2 message. The VTU-O and VTU-R continue to exchange O-CONTRACTn and the R-MARGINn messages until the VTU-O is satisfied and sends the O-B&G message which ends the negotiation.

576. A POSA would have understood that, throughout the exchange, the VTU-O would not propose a contract that exceeds the available interleaver memory at the VTU-R, which is described in the “maximum interleaver memory” field of the R-MSG2 sent by the VTU-R.

577. A POSA combining Mazzoni and VDSL1 would naturally use configuration messages, such as those taught in VDSL1 to agree on interleaver settings that achieve the best performance while not exceeding the capability of either transceiver. As discussed in the Technology Background in Section **Error! Reference source not found.**, a POSA would know that configuration messages are used, as a general practice, by communications systems to exchange capabilities and resource limitations and to agree on parameters so that the signals transmitted by one transceiver can be properly decoded by the other transceiver. A POSA would further know that when one transceiver learns, through a configuration message, from another transceiver about

its capabilities or its resource limitations that it must select parameters that allow the other transceiver to operate within its capabilities and within its resource limitations. As discussed below with respect to element b, memory allocations for interleavers and deinterleavers are determined based on the specific service selected and in particular whether it is a symmetric or asymmetric service. In combining Mazzoni with VDSL1, a POSA would know how to determine the maximum interleaver memory to be communicated in R-MSG2 based on, for example, whether the service to be provided is asymmetric or symmetric. For a symmetric service, half of the available shared memory will be used for the upstream interleaver. For an asymmetric service a much smaller amount is allocated for the upstream interleaver. For the example of six asymmetric services and six symmetric services described in Mazzoni, all of the asymmetric services have the same upstream rate of 32x64 kbit/s and would all require the same upstream interleaver memory allocation. Thus, for the example services provided in the Mazzoni specification, the maximum interleaver memory to be communicated in R-MSG2 can be determined from the knowledge of the whether the service is symmetric or asymmetric and the knowledge of the total size of the common memory that is shared by interleaving and deinterleaving.

578. VDSL1 provides a set of configurations messages that allow the VTU-O and VTU-R (called the TO and TU in Mazzoni) to exchange information about resource limitations and capabilities, and to agree on the parameters of the Reed-Solomon coding, interleaving, and other aspects of the upstream and downstream transmissions. A POSA would have known that such an exchange of information about resource limitations and capabilities and agreement on parameters is necessary, and as described above would be familiar with the use of configuration messages to reach an agreement on communication parameters in a way that does not exceed available resources and capabilities of the other transceiver. Thus, this use of configuration messages as taught by

VDSL1 would not even have been viewed as innovative in 2004. Nonetheless, VDSL1 provides a clear description of such a configuration message protocol.

579. VDSL1 teaches using configuration messages to communicate the maximum interleaver memory available at the VTU-R, for example through the “Maximal interleaver memory” field of R-MSG2. In the context of Mazzoni, which explicitly teaches the use of a single shared memory for interleaving and deinterleaving, a POSA would have understood that in combining the teachings of Mazzoni and VDSL1, a VTU-R transceiver practicing the shared-memory teachings of Mazzoni could in addition communicate the maximum amount of shared memory available at the VTU-R for interleaving and deinterleaving.

580. A POSA would have understood that in combining the teachings of Mazzoni and VDSL1, any transceiver, either a VTU-O or a VTU-R, following the teachings of Mazzoni and VDSL1 would preferably implement a shared memory to support interleaving in one direction and deinterleaving in the other direction. For example, a VTU-O would utilize a shared memory for downstream interleaving and upstream deinterleaving. As another example, a VTU-R would utilize a shared memory for upstream interleaving and downstream interleaving.

581. Such a VTU-O following the teachings of Mazzoni and VDSL1, would, of course, propose interleaver settings in the O-CONTRACTn messages that do not exceed the capabilities of the VTU-R. Therefore, the VTU-O will allocate a numbers of bytes for its deinterleaver/interleaver that do not exceed the maximum number of bytes specified in messages received from the VTU-R. Naturally, the VTU-O will allocate a numbers of bytes for its deinterleaver/interleaver that do not exceed its own shared memory resource.

582. Similarly, a VTU-R following the teachings of Mazzoni and VDSL1 would use configuration messages to communicate its capabilities to the VTU-O so that the VTU-O only

proposes interleaver settings in the O-CONTRACTn messages for which the upstream interleaver and downstream deinterleaver in the VTU-R do not exceed the VTU-R's available memory resources.

583. A POSA would have known that when a first transceiver learns, through a configuration message, from a second transceiver about its capabilities or its resource limitations that the first transceiver must select parameters that allow the second transceiver to operate within its capabilities and within its resource limitations. If the first transceiver selects parameter settings that exceed the communicated capabilities of the second transceiver, the system will not work. A POSA would have known this and avoided this obvious failure mode.

584. Of course, a natural combination of Mazzoni and VDSL1 would be to have both the VTU-R and the VTU-O implement transceivers that each use a single shared memory for interleaving and deinterleaving. As in the cases discussed above, a POSA combining the teachings of Mazzoni and VDSL1 would always ensure that the allocated memory for the deinterleaver/interleaver does not exceed the maximum resources at either transceiver.

585. A POSA having full view of the teachings of Mazzoni about the benefits of using shared memory for interleaving and deinterleaving and the disclosures in VDSL1 of configuration messages would ensure that the transceivers reach agreement on parameter settings that do not exceed the available interleaver memory at either transceiver. A POSA would be motivated to use configuration messages such as those in VDSL1 to reach agreement on parameter settings in the context of shared memory for interleaving and deinterleaving as described in Mazzoni.

586. As shown above, both Mazzoni and VDSL1 teach the use of transceivers. Mazzoni is directed towards a VDSL system that is adaptable in its choice of rates and interleavers. Such adaptability requires that the operator end (VTU-O) and user end (VTU-R) agree on the specific

and is intended to include anyone interested in and possessing ordinary skill regarding DSL standards. I also understand that, outside of receiving a membership, one may gain access to ITU documents as a guest of an ITU member.

321. Furthermore, the G.993.1 standard that was approved in June 2004 shares many concepts with the ETSI VDSL1 standard TS 101 270-2 that bears a copyright date of 2003 and was published on ETSI's website to the public at large in July 2003 (specifically July 24, 2003, according to the Work Programme details for the document available through the ETSI website: [https://portal.etsi.org/webapp/workprogram/Report\\_WorkItem.asp?WKI\\_ID=11828](https://portal.etsi.org/webapp/workprogram/Report_WorkItem.asp?WKI_ID=11828).) The shared concepts found in ETSI VDSL1 that are included in and relevant to my invalidity analysis include support and symmetric and asymmetric rates by a VDSL transceiver (e.g., ETSI VDSL1 § 5.2.5.1.1), the use of Reed-Solomon Coding (e.g., ETSI VDSL1 § 6.4.2), the use of convolutional interleaving (specifically the triangular implementation of convolutional interleaving) (e.g., ETSI VDSL1 §§ 6.4.3.1, 6.4.3.2), and the initialization process (including specifically the O-MSG2, R-MSG2, and "maximum interleaver memory" field used in my analysis) (e.g., ETSI VDSL1 §§ 8.2.1, 8.2.4, 8.2.6 (generally), 8.2.6.2.1.1, 8.2.6.2.1.2, 8.2.6.2.1.3, 8.2.6.3.1.1, 8.2.6.3.1.2, 8.2.6.3.1.3, 8.2.6.3.1.4). For these concepts, ETSI VDSL1 includes substantially the same teachings, and indeed often identical content for concepts like RS coding, interleaving, and the initialization process, as described in the version of ITU 993.1 that was approved in June 2004.

322. I note that the connection between ETSI's VDSL1 standard TS 101 270-2 and ITU VDSL standards was documented, for example, in TD-WP1-0002. TD-WP1-0002 is a letter associated with the April 2004 ITU meeting in Geneva from ETSI TM6 Chairman Manfred Gindel to Paolo Rosa of the ITU describing a compromise proposal that would "enable ITU-T SG15Q4 to develop a VDSL specification similar to TS 101 270-2, but with DMT in the main body and QAM

331. VDSL1 describes communications devices capable of transmitting and receiving data. For example, VDSL1 describes VDSL Transceiver Units (VTUs) that are capable of transmitting and receiving data. Specifically, the VTU at the remote site (VTU-R) is capable of transmitting data on the upstream channel and receiving data on the downstream channel. The VTU at the Optical Network Unit (ONU) is capable of transmitting data on the downstream channel and receiving data on the upstream channel. (See Section 4 Abbreviations of VDSL1.)

VTU	VDSL Transceiver Unit
VTU-O	VTU at the ONU
VTU-R	VTU at the Remote site
VTU-x	Any one of VTU-O or VTU-R

(VDSL1 at Section 4 Abbreviations, page 4)

332. It is a common practice well known by a POSA at the time of the invention for the transmitter and receiver portion of a communications device such as a DSL transmitter/receiver to share circuitry. At the very least, such devices share circuitry related to providing power. In typical transceiver architectures known to a POSA at the time of the invention, transmitting and receiving functions will additionally share a common processing resource such as a digital signal processing chip and/or a common memory resource. For example, a single digital signal processing chip was commonly used for both transmitting and receiving functions as discussed in the Technology Background section above. Both transmitting and receiving functions commonly share the same clock. Both transmitting and receiving functions commonly share the same antenna and associated analog circuitry. As another example, whenever the device utilizes shared memory for interleaving and deinterleaving, the device would of necessity share common circuitry related to the shared memory.



447. As discussed above, Mazzoni describes specifically how to determine the memory requirements for the interleavers and deinterleavers associated with a particular overall bit rate at, for example, Mazzoni 2:39-56. VDSL1 performs the same computations as exemplified, for example, by the description provided in section 8.4.2 including the equations given in Table 8-1 and the example calculations provided in Table 8-2.

448. VDSL1 provides a detailed descriptions of the configuration messages that it uses to adapt the interleaver settings and the bit rate, as discussed, for example, above in Section X.C.

11. A POSA would have known how to use configuration messages

449. As discussed in the Technology Background in Section **Error! Reference source not found.**, a POSA would know that configuration messages are used, as a general practice, by communications systems to exchange capabilities and resource limitations and to agree on parameters so that the signals transmitted by one transceiver can be properly decoded by the other transceiver. A POSA would further know that when one transceiver learns, through a configuration message, from another transceiver about its capabilities or its resource limitations that it must select parameters that allow the other transceiver to operate within its capabilities and within its resource limitations.

12. VDSL1 configuration messages work with Mazzoni

450. As described above, VDSL1 and Mazzoni are both directed to essentially the same VDSL communication system, which supports asymmetrical and symmetrical services using the same Reed-Solomon coding, the same convolutional interleaving with the same implementation of triangular interleaving using virtual shift registers implemented in a memory and even equivalent mathematical expressions to determine how much memory is required to implement the interleaving and de-interleaving in that memory.

451. VDSL1 provides a set of configurations messages that allow the VTU-O and VTU-R (called the TO and TU in Mazzoni) to exchange information about resource limitations and capabilities, and to agree on the parameters of the Reed-Solomon coding, interleaving, and other aspects of the upstream and downstream transmissions. A POSA would have known that such an exchange of information about resource limitations and capabilities and agreement on parameters is necessary, and as described above would be familiar with the use of configuration messages to reach an agreement on communication parameters in a way that does not exceed available resources and capabilities of the other transceiver. Thus, this use of configuration messages as taught by VDSL1 would not even have been viewed as innovative in 2004. Nonetheless, VDSL1 provides a clear description of such a configuration message protocol.

452. VDSL1 teaches using configuration messages to communicate the maximum interleaver memory available at the VTU-R, for example through the “Maximal interleaver memory” field of R-MSG2. In the context of Mazzoni, which explicitly teaches the use of a single shared memory for interleaving and deinterleaving, a POSA would have understood that in combining the teachings of Mazzoni and VDSL1, a VTU-R transceiver practicing the shared-memory teachings of Mazzoni could in addition communicate the maximum amount of shared memory available at the VTU-R for interleaving and deinterleaving.

453. A POSA would have understood that in combining the teachings of Mazzoni and VDSL1, any transceiver, either a VTU-O or a VTU-R, following the teachings of Mazzoni and VDSL1 would preferably implement a shared memory to support interleaving in one direction and deinterleaving in the other direction. For example, a VTU-O would utilize a shared memory for downstream interleaving and upstream deinterleaving. As another example, a VTU-R would utilize a shared memory for upstream interleaving and downstream interleaving.

511. Fadavi-Ardekani uses the phrases “headend” transceiver and “central office” transceiver to describe the transceiver called a VTU-O in VDSL1. See, for example, 3:5-8. However, neither the word “headend” nor the word “central” nor the word “office” appear in any of the 26 claims of Fadavi-Ardekani. Thus a POSA would have understood the claims and invention of Fadavi-Ardekani to apply equally to the VTU-O and the VTU-R. I note that a business customer at the time of the asserted invention would commonly have multiple twisted-pair lines terminating at a business location (see, e.g., <https://www.prweb.com/releases/2003/11/prweb91311.htm>) and a POSA would know that Fadavi-Ardekani can be implemented at the business location to efficiently support DSL sessions on each twisted pair line terminating at the business location.

4. VDSL1 provide benefits to Fadavi Ardekani

512. Fadavi-Ardekani and VDSL1 are documents that are in the same time period address the same specific application of DSL communications and provide complementary ideas that benefit each other significantly. Fadavi-Ardekani teaches flexible memory sharing to provide the capability of adapting to support various upstream and downstream data rates with the least possible amount of memory. Fadavi-Ardekani focuses on the transceiver implementation associated with such a system rather than the configuration messages that would be sent and received during the initialization of such a system. VDSL1 complements and benefits Fadavi-Ardekani by providing those configuration messages and in particular specific configuration messages to communicate the capabilities available for interleaving/deinterleaving, such as maximum interleaver memory. An additional benefit from VDSL1 is a description of how to communicate explicit interleaver settings in configuration messages sent during initialization. Combining the concepts of these two documents leads to a better overall system that is efficient in its use of memory, supports a wide range of possible interleaver settings, and selects interleaver settings that, naturally, do not exceed the memory constraints of either transmitter.

2. The device according to claim 1 wherein the size of the first memory space is equal to  $I \times (I-1) \times M/2$  bytes, the size of the second memory space is equal to  $I' \times (I'-1) \times M'/2$  bytes, and the sizes of the first and second memory spaces are set by I, I', M and M'.

537. Moreover, and as a result of the specific VDSL system and in view of the specific disclosures from Mazzoni above, the POSA would have understood that Mazzoni discloses common memory used by at least two functions.

538. VDSL1 describes a transceiver as construed by the court as well. VDSL1 describes a communications devices capable of transmitting and receiving data. For example, VDSL1 describes VDSL Transceiver Units (VTUs) that are capable of transmitting and receiving data. Specifically, the VTU at the remote site (VTU-R) is capable of transmitting data on the upstream channel and receiving data on the downstream channel. The VTU at the Optical Network Unit (ONU) is capable of transmitting data on the downstream channel and receiving data on the upstream channel. (See Section 4 Abbreviations of VDSL1.)

VTU	VDSL Transceiver Unit
VTU-O	VTU at the ONU
VTU-R	VTU at the Remote site
VTU-x	Any one of VTU-O or VTU-R

(VDSL1 at Section 4 Abbreviations, page 4)

539. It is a common practice well known by a POSA at the time of the invention for the transmitter and receiver portion of a communications device such as a DSL transmitter/receiver to share circuitry. At the very least, such devices share circuitry related to providing power. In typical transceiver architectures known to a POSA at the time of the invention, transmitting and receiving functions will additionally share a common processing resource such as a digital signal processing chip and/or a common memory resource. For example, a single digital signal processing chip was

commonly used for both transmitting and receiving functions as discussed in the Technology Background section above. Both transmitting and receiving functions commonly share the same clock. Both transmitting and receiving functions commonly share the same antenna and associated analog circuitry. As another example, whenever the device utilizes shared memory for interleaving and deinterleaving, the device would of necessity share common circuitry related to the shared memory.

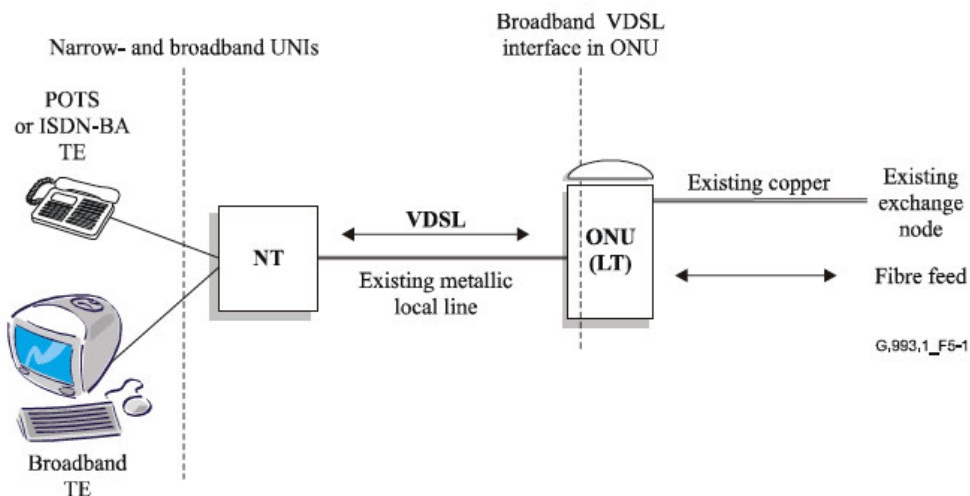
540. As generally understood by a POSA and as described by Mazzoni, a VDSL system is a digital communication system linking an operator and users via very high bit rate transmission lines. Mazzoni further discloses that the invention applies to the send/receive devices at both the operator and user ends of a transmission line. Particularly, Mazzoni explains that “the invention applies to a digital communication system linking an operator and users via very high bit rate transmission lines. Thus, the invention applies more particularly to send/receive devices, usually referred to as modems, at the operator and user ends of a transmission line.” Mazzoni. at 1:22-27.

541. In a VDSL system, data is transmitted in both directions between two send/receive devices. That is, the modem at the operator end sends data to the user end and receives data from the user end. Similarly, the modem at the user end sends data to the operator end and receives data from the operator end. In VDSL, the data transmission from the operator end to the receiver end is often called the *downstream transmission* or *downlink* and the data transmission from the receiver end to the operator end is often called the *upstream transmission* or *uplink*.

542. A driving motivation for the memory sharing described in Mazzoni is that the upstream transmission and the downstream transmission may be configured differently according to a variety of possible “services”, which requires that the interleaver and deinterleaver also be configurable in terms of their memory allocations. Depending on the specific upstream and

## 5.1 General reference models

Figure 5-1 shows the reference configuration used for G.993.1. It is essentially a Fibre to the Node architecture with an Optical Network Unit (ONU) sited in the existing metallic access network (or at the serving Local Exchange or Central Office). The first architectural model covers Fibre-to-the-cabinet (FTTCab) type of deployment; the second one is Fibre-to-the-exchange (FTTEx) type of deployment. Existing unscreened twisted metallic access wire-pairs are used to convey the signals to and from the customer's premises.



**Figure 5-1/G.993.1 – General reference model**

561. The transmitter and receiver portions of the VTUs share some circuitry. For example, such devices share the circuitry that provides power. Also, the configuration messages that are received by the VTU are used to affect both transmitter and receiver functionality, so the circuitry that utilizes the messages is shared by the transmitter and receiver portions of the VTU. The transmitting and receiving functions commonly share the same clock to control the timing of the digital circuitry. As another example, whenever the VTU utilizes shared memory for interleaving and deinterleaving, as taught by Mazzoni, the transmitter portion and receiver portion of the VTU share at least the common circuitry of the common memory used for interleaving and deinterleaving.

562. It is a common practice well known by a POSA at the time of the invention for the transmitter and receiver portion of a communications device such as a DSL transmitter/receiver to

share circuitry. At the very least, such devices share circuitry related to providing power. In typical transceiver architectures known to a POSA at the time of the invention, transmitting and receiving functions will additionally share a common processing resource such as a digital signal processing chip and/or a common memory resource. For example, a single digital signal processing chip was commonly used for both transmitting and receiving functions as discussed in the Technology Background section above. Both transmitting and receiving functions commonly share the same clock. Both transmitting and receiving functions commonly share the same antenna and associated analog circuitry. As another example, whenever the device utilizes shared memory for interleaving and deinterleaving, the device would of necessity share common circuitry related to the shared memory.

563. In addition to disclosing a transceiver, the VDSL1 standard discloses transmitting or receiving a message during initialization specifying a maximum number of bytes of memory that are available to be allocated to an interleaver. Specifically, the VDSL1 standard discloses how each of its transceivers transmit or receive a message during initialization specifying a maximum number of bytes of memory that are available to be allocated to an interleaver in the “channel analysis and exchange state” of the “initialization protocol.” VDSL1 provides a complete protocol of configuration messages including a message during initialization specifying a maximum number of bytes of memory that are available to be allocated to an interleaver, as described above in Section X.C.8 and included here by reference.

564. Figure 12-2 from VDSL1, shown below, shows the “Channel analysis and exchange” state as part of initialization.



CONTRACT1. A POSA would have understood that the upstream contract proposed by the VTU-O must comport with the limitations of the VTU-R including not exceeding the maximum interleaver memory as communicated in R-MSG2.

575. When the VTU-R receives the O-CONTRACT1 message, it computes the minimal SNR margin over all tones and sends this information to the VTU-O in the R-MARGIN1 message. If the VTU-O finds the SNR margins in the R-MARGIN1 message acceptable, then it sends the O-B&G message, which ends the negotiation. If the VTU-O does not find the SNR margins in the R-MARGIN1 message acceptable, then it sends a new O-CONTRACT2 message and the VTU-R computes the new values for the minimal SNR margin over all tones and sends this information to the VTU-O in the R-MARGIN2 message. The VTU-O and VTU-R continue to exchange O-CONTRACTn and the R-MARGINn messages until the VTU-O is satisfied and sends the O-B&G message which ends the negotiation.

576. A POSA would have understood that, throughout the exchange, the VTU-O would not propose a contract that exceeds the available interleaver memory at the VTU-R, which is described in the “maximum interleaver memory” field of the R-MSG2 sent by the VTU-R.

577. A POSA combining Mazzoni and VDSL1 would naturally use configuration messages, such as those taught in VDSL1 to agree on interleaver settings that achieve the best performance while not exceeding the capability of either transceiver. As discussed in the Technology Background in Section **Error! Reference source not found.**, a POSA would know that configuration messages are used, as a general practice, by communications systems to exchange capabilities and resource limitations and to agree on parameters so that the signals transmitted by one transceiver can be properly decoded by the other transceiver. A POSA would further know that when one transceiver learns, through a configuration message, from another transceiver about

its capabilities or its resource limitations that it must select parameters that allow the other transceiver to operate within its capabilities and within its resource limitations. As discussed below with respect to element b, memory allocations for interleavers and deinterleavers are determined based on the specific service selected and in particular whether it is a symmetric or asymmetric service. In combining Mazzoni with VDSL1, a POSA would know how to determine the maximum interleaver memory to be communicated in R-MSG2 based on, for example, whether the service to be provided is asymmetric or symmetric. For a symmetric service, half of the available shared memory will be used for the upstream interleaver. For an asymmetric service a much smaller amount is allocated for the upstream interleaver. For the example of six asymmetric services and six symmetric services described in Mazzoni, all of the asymmetric services have the same upstream rate of 32x64 kbit/s and would all require the same upstream interleaver memory allocation. Thus, for the example services provided in the Mazzoni specification, the maximum interleaver memory to be communicated in R-MSG2 can be determined from the knowledge of the whether the service is symmetric or asymmetric and the knowledge of the total size of the common memory that is shared by interleaving and deinterleaving.

578. VDSL1 provides a set of configurations messages that allow the VTU-O and VTU-R (called the TO and TU in Mazzoni) to exchange information about resource limitations and capabilities, and to agree on the parameters of the Reed-Solomon coding, interleaving, and other aspects of the upstream and downstream transmissions. A POSA would have known that such an exchange of information about resource limitations and capabilities and agreement on parameters is necessary, and as described above would be familiar with the use of configuration messages to reach an agreement on communication parameters in a way that does not exceed available resources and capabilities of the other transceiver. Thus, this use of configuration messages as taught by

VDSL1 would not even have been viewed as innovative in 2004. Nonetheless, VDSL1 provides a clear description of such a configuration message protocol.

579. VDSL1 teaches using configuration messages to communicate the maximum interleaver memory available at the VTU-R, for example through the “Maximal interleaver memory” field of R-MSG2. In the context of Mazzoni, which explicitly teaches the use of a single shared memory for interleaving and deinterleaving, a POSA would have understood that in combining the teachings of Mazzoni and VDSL1, a VTU-R transceiver practicing the shared-memory teachings of Mazzoni could in addition communicate the maximum amount of shared memory available at the VTU-R for interleaving and deinterleaving.

580. A POSA would have understood that in combining the teachings of Mazzoni and VDSL1, any transceiver, either a VTU-O or a VTU-R, following the teachings of Mazzoni and VDSL1 would preferably implement a shared memory to support interleaving in one direction and deinterleaving in the other direction. For example, a VTU-O would utilize a shared memory for downstream interleaving and upstream deinterleaving. As another example, a VTU-R would utilize a shared memory for upstream interleaving and downstream interleaving.

581. Such a VTU-O following the teachings of Mazzoni and VDSL1, would, of course, propose interleaver settings in the O-CONTRACTn messages that do not exceed the capabilities of the VTU-R. Therefore, the VTU-O will allocate a numbers of bytes for its deinterleaver/interleaver that do not exceed the maximum number of bytes specified in messages received from the VTU-R. Naturally, the VTU-O will allocate a numbers of bytes for its deinterleaver/interleaver that do not exceed its own shared memory resource.

582. Similarly, a VTU-R following the teachings of Mazzoni and VDSL1 would use configuration messages to communicate its capabilities to the VTU-O so that the VTU-O only

(ONU) is capable of transmitting data on the downstream channel and receiving data on the upstream channel. (See Section 4 Abbreviations of VDSL1.)

VTU	VDSL Transceiver Unit
VTU-O	VTU at the ONU
VTU-R	VTU at the Remote site
VTU-x	Any one of VTU-O or VTU-R

VDSL1 at Section 4 Abbreviations, page 4.

624. It is a common practice well known by a POSA at the time of the invention for the transmitter and receiver portion of a communications device such as a DSL transmitter/receiver to share circuitry. At the very least, such devices share circuitry related to providing power. In typical transceiver architectures known to a POSA at the time of the invention, transmitting and receiving functions will additionally share a common processing resource such as a digital signal processing chip and/or a common memory resource. For example, a single digital signal processing chip was commonly used for both transmitting and receiving functions as discussed in the Technology Background section above. Both transmitting and receiving functions commonly share the same clock. Both transmitting and receiving functions commonly share the same antenna and associated analog circuitry. As another example, whenever the device utilizes shared memory for interleaving and deinterleaving, the device would of necessity share common circuitry related to the shared memory.

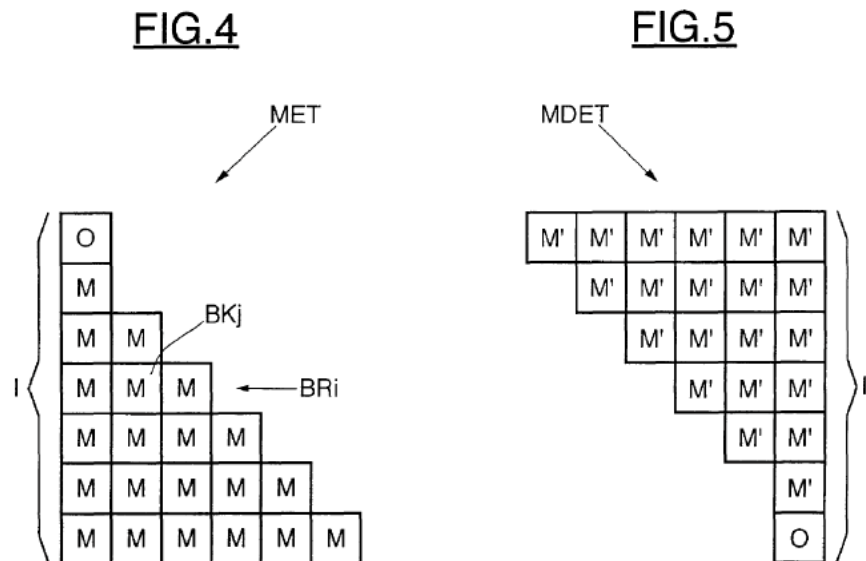
625. Both Mazzoni and VDSL1 disclose interleaving of Reed-Solomon coded bytes. Mazzoni describes the interleaving of Reed-Solomon coded bytes, for example, at 4:36-41 as follows:

The channel coding unit CC includes Reed-Solomon coding means whose structure and function are known to those of skill in the art. The Reed-Solomon coding means are associated with the interleaving means. In conjunction with subsequent interleaving, the

664. Mazzoni describes the same memory allocation for the same the triangular implementation of convolutional interleaving as VDSL1. In fact, even the illustrations of Mazzoni and VDSL1 are structurally identical. Mazzoni describes the triangular implementation of convolutional interleaving for example at 5:9-20.

The internal architecture and the operation of the interleaving and deinterleaving means will now be described in more detail with more particular reference to FIGS. 3-8. As shown in FIG. 3, and as already explained, the interleaving means MET follow the Reed-Solomon coding means CRS, and the deinterleaving means MDET precede the ReedSolomon decoding means DCRS. As shown diagrammatically in FIGS. 4 and 5, the interleaving and deinterleaving are convolutional triangular interleaving and deinterleaving. There are I branches of i-1 blocks of M bytes for interleaving and I' branches of i'-1 blocks of M' bytes for deinterleaving. (Mazzoni 5:9-20)

665. Figures 4 and 5 on drawing sheet 4 of Mazzoni illustrate the triangular implementation of convolutional interleaving, which is the same implementation as Fig. 8-2 of VDSL1.



memory space assigned to said interleaver and a second memory space assigned to said deinterleaver, a size of each of the first and second memory spaces being set as a function of the bit rate actually processed by the device,

a Reed-Solomon coder/decoder connected to said interleaver and said deinterleaver and having a length  $N$ , and

said interleaver providing convolutional interleaving of  $I$  branches with  $i-1$  blocks of  $M$  bytes, and said deinterleaver providing convolutional deinterleaving with  $I'$  branches of  $i'-1$  blocks of  $M'$  bytes, with  $I$  and  $I'$  being sub-multiples of  $N$  and  $i$  and  $i'$  being current relative indexes of the branches.

(Mazzoni 8:12-24)

675. Claim 1 explicitly describes shared memory with the words “a shared memory having...” and allocating a first number of bytes of the shared memory to the deinterleaver/interleaver for reception/transmission at a first data rate with the words “having a first memory space assigned to said interleaver and a second memory space assigned to said deinterleaver, a size of each of the first and second memory spaces being set as a function of the bit rate actually processed by the device.”

676. Claim 1 explicitly describes that the deinterleaving/interleaving is deinterleaving/interleaving of a first plurality of Reed Solomon (RS) coded data bytes with the words “a Reed-Solomon coder/decoder connected to said interleaver and said deinterleaver.”

677. Claim 8 has similar language to claim 1. (See Mazzoni 9:9-46.) Claim 13 of Mazzoni also discloses allocating a first memory space for interleaving and a second memory space for deinterleaving in a shared memory for interleaving and deinterleaving of Reed-Solomon coded bytes. (See Mazzoni 10:22-42)

678. VDSL1 teaches using a message sent during initialization to communicate the maximum amount of memory available for interleaving/deinterleaving. As discussed in the Technology Background in Section **Error! Reference source not found.**, a POSA at the time of

the invention would have known that configuration messages are used, as a general practice, by communications systems to exchange capabilities and resource limitations and to agree on parameters so that the signals transmitted by one transceiver can be properly decoded by the other transceiver. A POSA would further know that when one transceiver learns, through a configuration message, from another transceiver about its capabilities or its resource limitations that it must select parameters that allow the other transceiver to operate within its capabilities and within its resource limitations. If one transceiver selects parameter settings that exceed the communicated capabilities of the other transceiver, the system will not work. A POSA would have known to avoid this obvious failure mode.

679. A POSA combining Mazzoni and VDSL1 would naturally use configuration messages, such as those taught in VDSL1 to agree on interleaver settings that achieve the best performance while not exceeding the capability of either transceiver. VDSL1 provides a set of configurations messages that allow the VTU-O and VTU-R (called the TO and TU in Mazzoni) to exchange information about resource limitations and capabilities, and to agree on the parameters of the Reed-Solomon coding, interleaving, and other aspects of the upstream and downstream transmissions.

680. VDSL1 teaches using configuration messages to communicate the maximum deinterleaver/interleaver memory available at the VTU-R, for example through the “Maximal interleaver memory” field of R-MSG2. A POSA at the time of the inventions would have known that the VTU-O and the VTU-R must agree on interleaver settings that will not exceed the capability of the VTU-R as communicated, for example, in the “Maximal interleaver memory” field by the VTU-R to the VTU-O. As a result, the VTU-O and VTU-R will, of course, allocate numbers of



1027. Mazzoni further describes the allocation of common memory in its claims. For example, claim 1 recites the following:

1. A device for sending and receiving digital data that is capable of processing different bit rates from a group of predetermined bit rates, the device comprising:

a channel coding/decoding stage comprising

an interleaver,

a deinterleaver,

a shared memory having a minimum size based upon a maximum bit rate of the group of predetermined bit rates and having a first memory space assigned to said interleaver and a second memory space assigned to said deinterleaver, a size of each of the first and second memory spaces being set as a function of the bit rate actually processed by the device,

a Reed-Solomon coder/decoder connected to said interleaver and said deinterleaver and having a length N,

and

said interleaver providing convolutional interleaving of I branches with i-1 blocks of M bytes, and said deinterleaver providing convolutional deinterleaving with I' branches of i'-1 blocks of M' bytes, with I and I' being sub-multiples of N and i and i' being current relative indexes of the branches.

*Id.* at 8:12–33.

1028. Claim 1 of Mazzoni discloses a shared memory where the size of the first and second memory spaces are set as a function of the bit rate actually processed by the device. Similarly, claim 2 provides the specific amount of memory set for a particular bit rate:

2. The device according to claim 1 wherein the size of the first memory space is equal to  $I \times (I-1) \times M/2$  bytes, the size of the second memory space is equal to  $I' \times (I'-1) \times M'/2$  bytes, and the sizes of the first and second memory spaces are set by I, I', M and M'.

1029. It is a common practice well known by a POSA at the time of the invention for the transmitter and receiver portion of a communications device such as a DSL transmitter/receiver to

share circuitry. At the very least, such devices share circuitry related to providing power. In typical transceiver architectures known to a POSA at the time of the invention, transmitting and receiving functions will additionally share a common processing resource such as a digital signal processing chip and/or a common memory resource. For example, a single digital signal processing chip was commonly used for both transmitting and receiving functions as discussed in the Technology Background section above. Both transmitting and receiving functions commonly share the same clock. Both transmitting and receiving functions commonly share the same antenna and associated analog circuitry. As another example, whenever the device utilizes shared memory for interleaving and deinterleaving, the device would of necessity share common circuitry related to the shared memory.

1030. LB-031 discloses a transceiver in a VDSL2 system. As generally understood by a POSA and as described by Mazzoni, a VDSL system is a digital communication system linking an operator and users via very high bit rate transmission lines. Mazzoni further discloses that the invention applies to the send/receive devices at both the operator and user ends of a transmission line. Particularly, Mazzoni explains that “the invention applies to a digital communication system linking an operator and users via very high bit rate transmission lines. Thus, the invention applies more particularly to send/receive devices, usually referred to as modems, at the operator and user ends of a transmission line.” Mazzoni. at 1:22-27.

1031. In a VDSL system, data is transmitted in both directions between two send/receive devices. That is, the modem at the operator end sends data to the user end and receives data from the user end. Similarly, the modem at the user end sends data to the operator end and receives data from the operator end. In VDSL, the data transmission from the operator end to the receiver end is

respective memory spaces assigned to the interleaving means and to the deinterleaving means. This is done according to the bit rate of the information sent by the terminal TO (parameters I and M) and the bit rate of the information received by the terminal TO (parameters I' and M'). (Mazzoni 5:21-30)

The above calculation of I, M, I' and M' for the asymmetrical service A6 can be applied in an analogous manner to the other services of the VDSL system. A table of values for the parameters I, M, I' and M' can therefore be stored in the coding/decoding stage. When the modem is installed at the end of the line, and depending on the service actually provided by the operator, the control means MCD may retrieve the corresponding values of I, M, I' and M' from the stored table. These values are delivered to the addressing means MAD1 and MAD2, the structure of which is described in more detail with reference to FIGS. 7 and 8. (Mazzoni 6:51-59)

1050. Such adaptability requires that the operator end (VTU-O) and user end (VTU-R) agree on the specific parameters that define those rates and interleavers. A POSA would have understood that for the communication system to work, each transceiver must know what rate is being transmitted by the other transceiver and what interleaving parameters are being used by the other transceiver. A POSA would have understood that this knowledge is required to correctly configure the transceiver. A POSA in June 2004 would have recognized, even without seeing Mazzoni and LB-031, that a common solution to ensuring that both transceivers know these configuration parameters would be to use configuration messages. Furthermore, A POSA in view of both Mazzoni and LB-031 would naturally use the shared memory of Mazzoni to adopt a range of rates and interleaver settings made available in VDSL1 (perhaps limited by the total shared memory available for that transceiver) and use configuration messages as disclosed in VDSL1 to agree on specific settings that do not exceed the capabilities of either transceiver.

1051. To the extent it is argued that Mazzoni does not disclose a transceiver that performs transmitting or receiving a message during initialization specifying a maximum number of bytes of memory that are available to be allocated to an interleaver, the LB-031 contribution does.

deinterleaver, a size of each of the first and second memory spaces being set as a function of the bit rate actually processed by the device.”

1115. Claim 1 explicitly describes that the deinterleaving/interleaving is deinterleaving/interleaving of a first plurality of Reed Solomon (RS) coded data bytes with the words “a Reed-Solomon coder/decoder connected to said interleaver and said deinterleaver.”

1116. Claim 8 has similar language to claim 1. (See Mazzoni 9:9-46.) Claim 13 of Mazzoni also discloses allocating a first memory space for interleaving and a second memory space for deinterleaving in a shared memory for interleaving and deinterleaving of Reed-Solomon coded bytes. (See Mazzoni 10:22-42)

1117. A POSA at the time of the invention would have known that configuration messages are used, as a general practice, by communications systems to exchange capabilities and resource limitations and to agree on parameters so that the signals transmitted by one transceiver can be properly decoded by the other transceiver.

1118. A POSA would further have known that when one transceiver learns, through a configuration message, from another transceiver about its capabilities or its resource limitations that it must select parameters that allow the other transceiver to operate within its capabilities and within its resource limitations. If one transceiver selects parameter settings that exceed the communicated capabilities of the other transceiver, the system will not work. A POSA would have known this, and would have avoided this obvious failure mode.

1119. LB-031 teaches the exchange of capabilities with one alternative being the maximum number of bytes of memory. A POSA would have known that the transceiver must choose interleaver settings that do not exceed the capabilities communicated in such a message.

VTU	VDSL Transceiver Unit
VTU-O	VTU at the ONU
VTU-R	VTU at the Remote site
VTU-x	Any one of VTU-O or VTU-R

(VDSL1 at Section 4 Abbreviations, page 4)

1384. It is a common practice well known by a POSA at the time of the invention for the transmitter and receiver portion of a communications device such as a DSL transmitter/receiver to share circuitry. At the very least, such devices share circuitry related to providing power. In typical transceiver architectures known to a POSA at the time of the invention, transmitting and receiving functions will additionally share a common processing resource such as a digital signal processing chip and/or a common memory resource. For example, a single digital signal processing chip was commonly used for both transmitting and receiving functions as discussed in the Technology Background section above. Both transmitting and receiving functions commonly share the same clock. Both transmitting and receiving functions commonly share the same antenna and associated analog circuitry. As another example, whenever the device utilizes shared memory for interleaving and deinterleaving, the device would of necessity share common circuitry related to the shared memory.

1385. A shared memory as construed by the court is “common memory used by at least two functions, where a portion of the memory can be used by either one of the functions.” The Interleave/De-Interleave Memory (IDIM) described in Fadavi-Ardekani meets the court’s construction of a shared memory. It is a common memory used by at least the two functions of interleaving and deinterleaving.